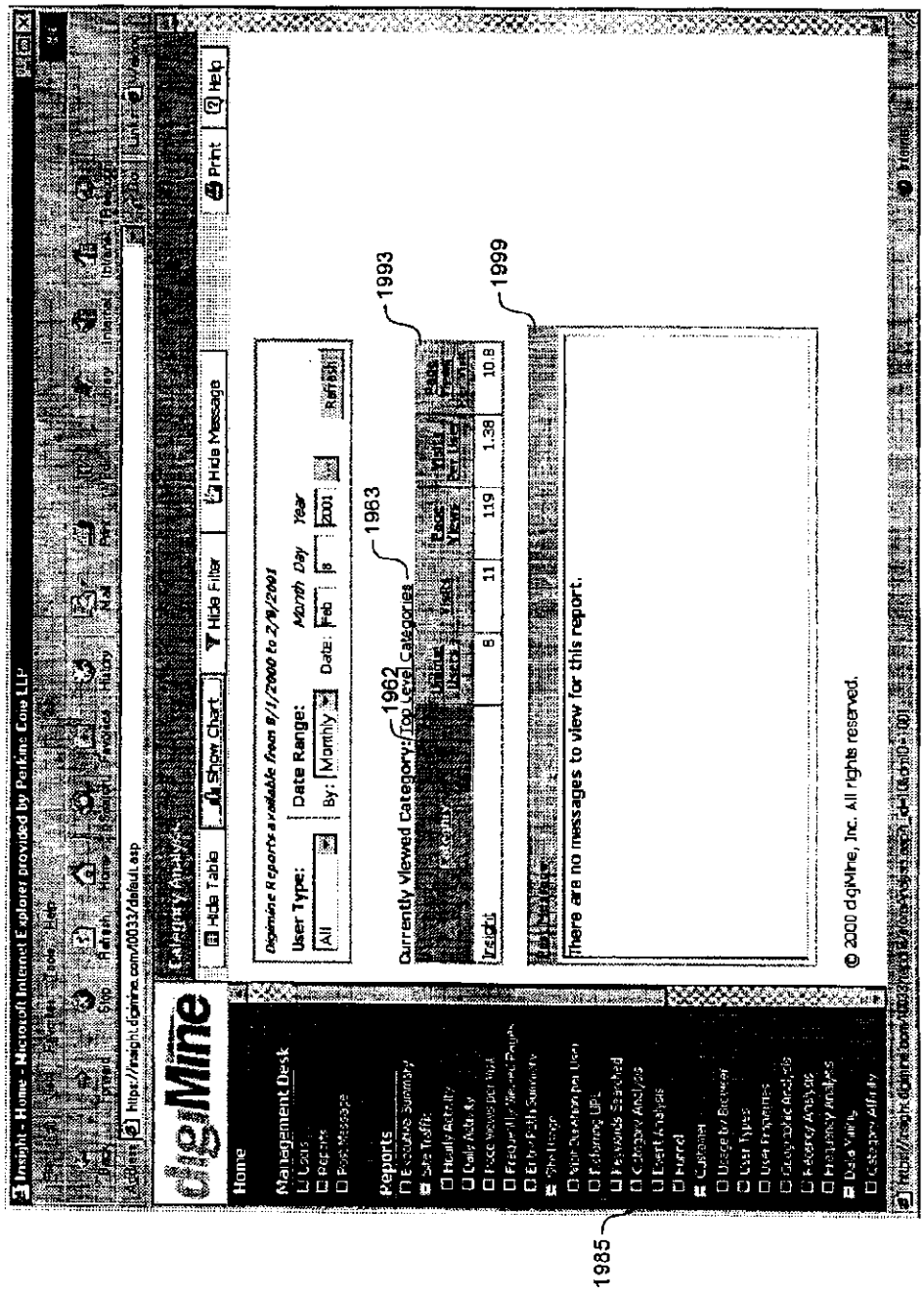

Fig. 19Z

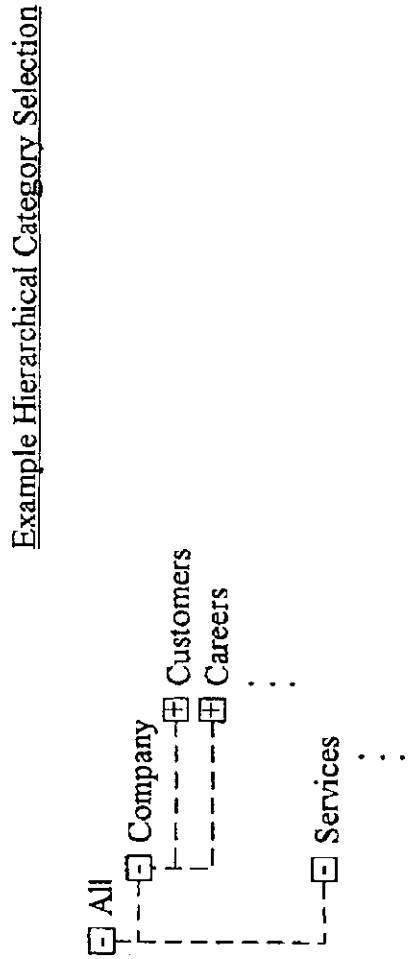
Example Hierarchical Category Selection
All
Company
Customers
Careers
. . .
Services
. . .

Fig. 19AA

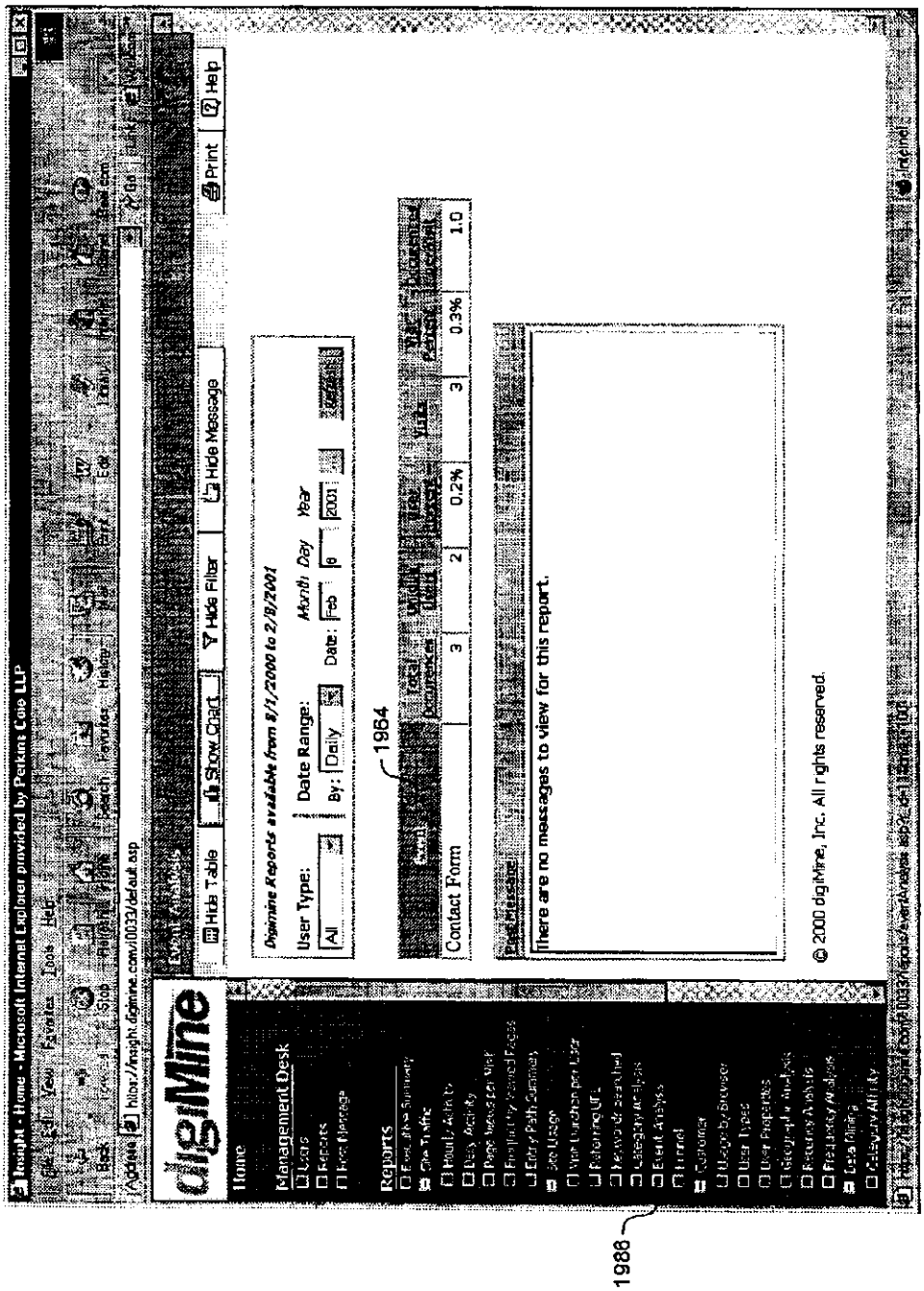
1984
1988
digiMine
Home
Management Desk
Reports
Hide Table
Show Chart
Hide Filter
Hide Message
Print
Help
User Type:
All
Date Range:
By: Daily
Month Day Year
Date: Feb 8 2001
Contact Form
Post Message
There are no messages to view for this report.
© 2000 digiMine, Inc. All rights reserved.

Fig. 19AB

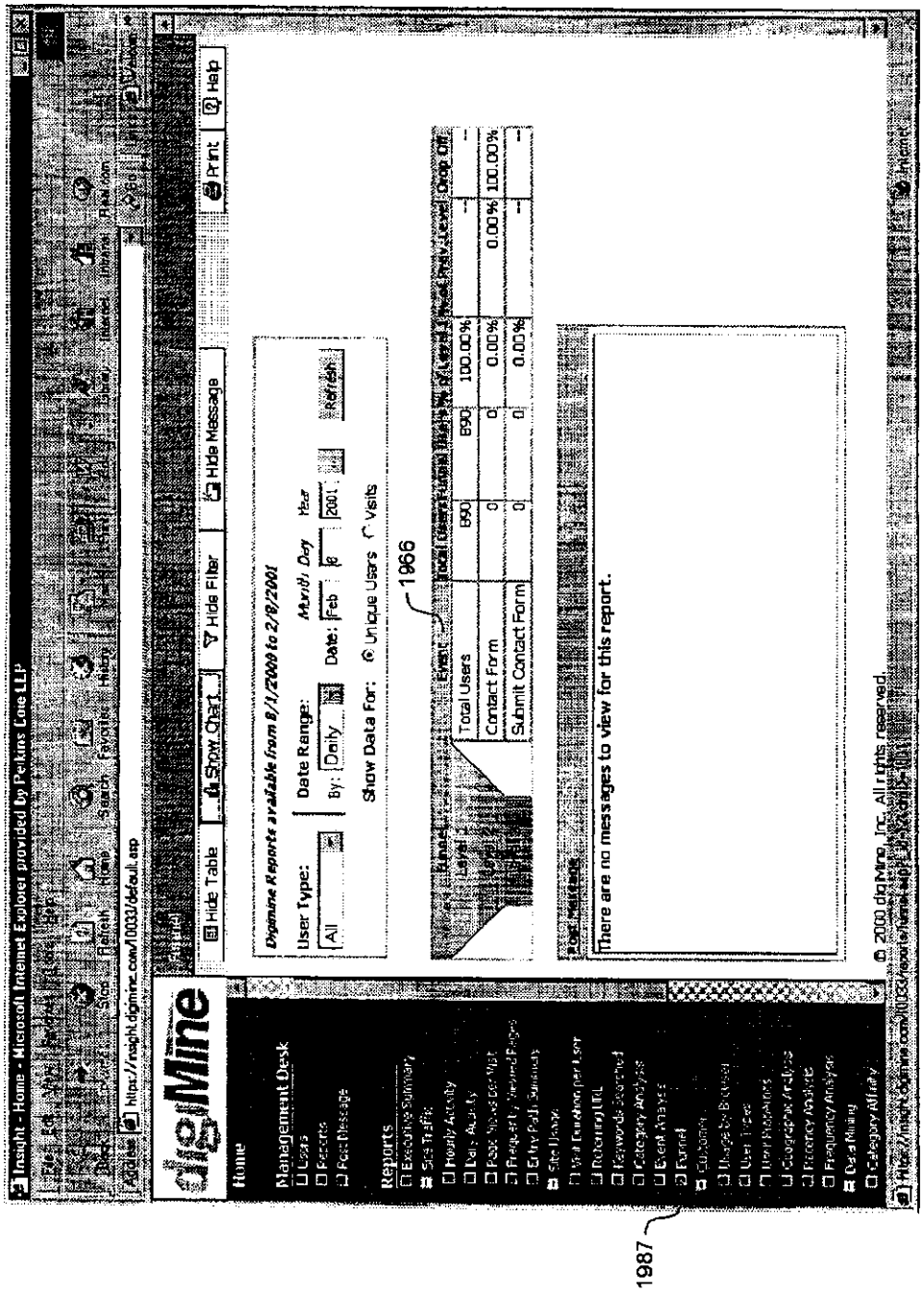

Fig. 19AC

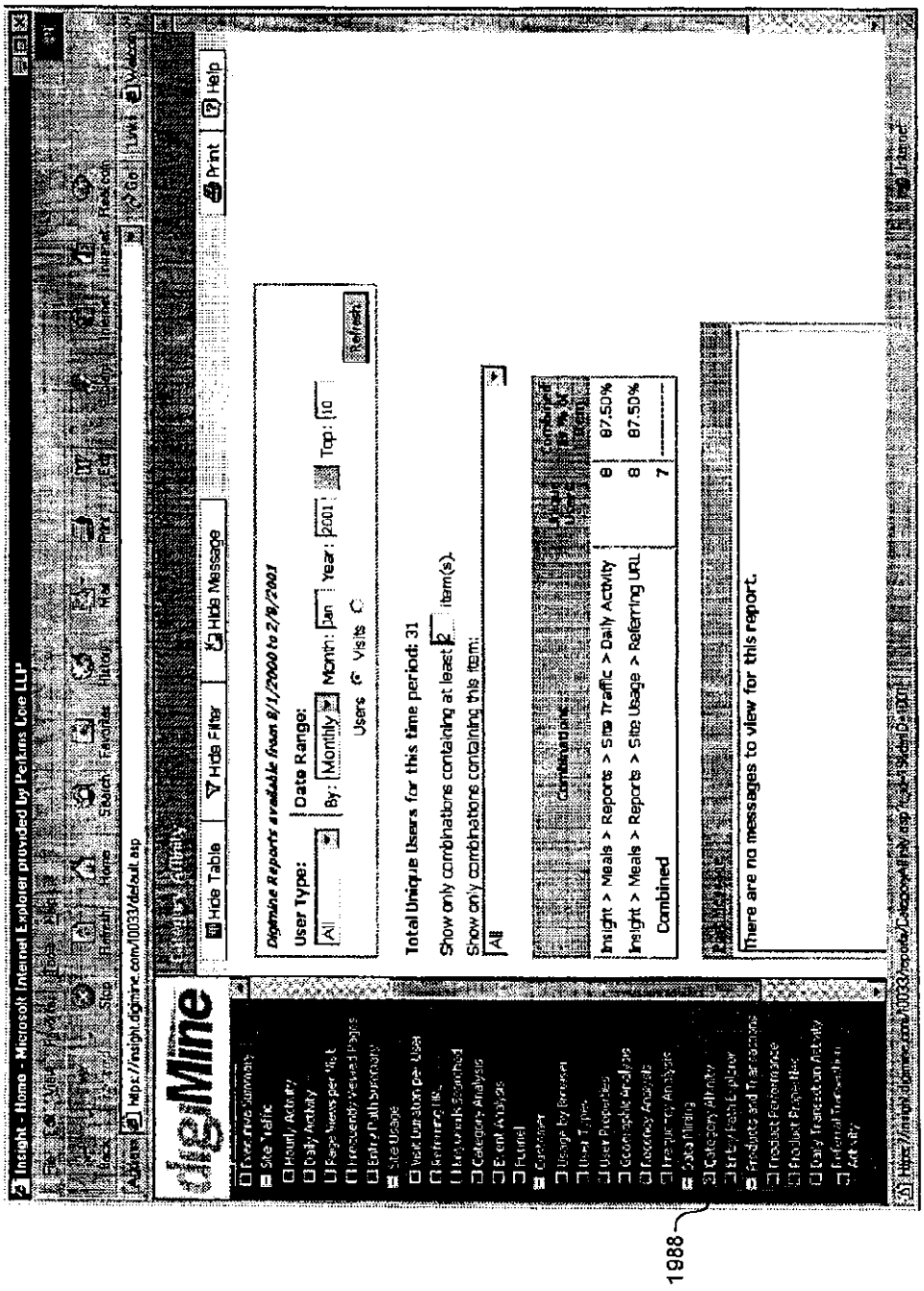
Insight - Home - Microsoft Internet Explorer provided by Perkins Coie LLP
DigiMine
Hide Table
Hide Filter
Hide Message
Print
Help
Digmine Reports available from 8/1/2000 to 2/8/2001
User Type:
Date Range:
By: Monthly
Month: Jan
Year: 2001
Top: 10
Refresh
Users
Visits
Total Unique Users for this time period: 31
Show only combinations containing at least 2 item(s).
Show only combinations containing this item:
All
Insight > Meals > Reports > Site Traffic > Daily Activity 8 87.50%
Insight > Meals > Reports > Site Usage > Referring URL 8 87.50%
Combined 7
There are no messages to view for this report.
1988

Fig. 19AD

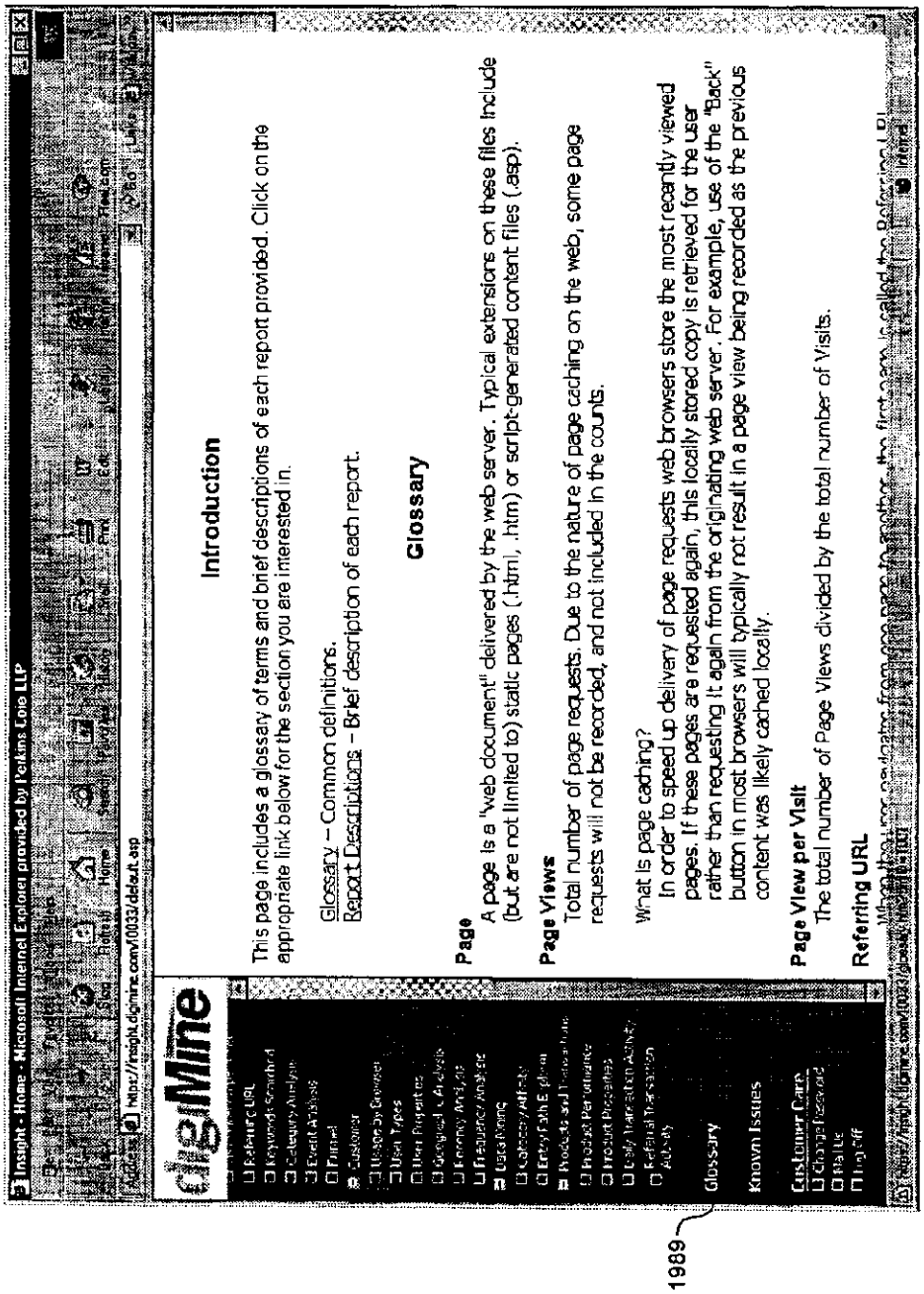
Insight - Home - Microsoft Internet Explorer provided by Perkins Coie LLP
https://insight.clicmine.com/0033/default.asp
clicMine
Glossary
Known Issues
Introduction
This page includes a glossary of terms and brief descriptions of each report provided. Click on the appropriate link below for the section you are interested in.
Glossary – Common definitions.
Report Descriptions - Brief description of each report.
Glossary
Page
A page is a "web document" delivered by the web server. Typical extensions on these files include (but are not limited to) static pages (.html, .htm) or script-generated content files (.asp).
Page Views
Total number of page requests. Due to the nature of page caching on the web, some page requests will not be recorded, and not included in the counts.
What is page caching?
In order to speed up delivery of page requests web browsers store the most recently viewed pages. If these pages are requested again, this locally stored copy is retrieved for the user rather than requesting it again from the originating web server. For example, use of the "Back" button in most browsers will typically not result in a page view being recorded as the previous content was likely cached locally.
Page View per Visit
The total number of Page Views divided by the total number of Visits.
Referring URL
1989

Fig. 19AE

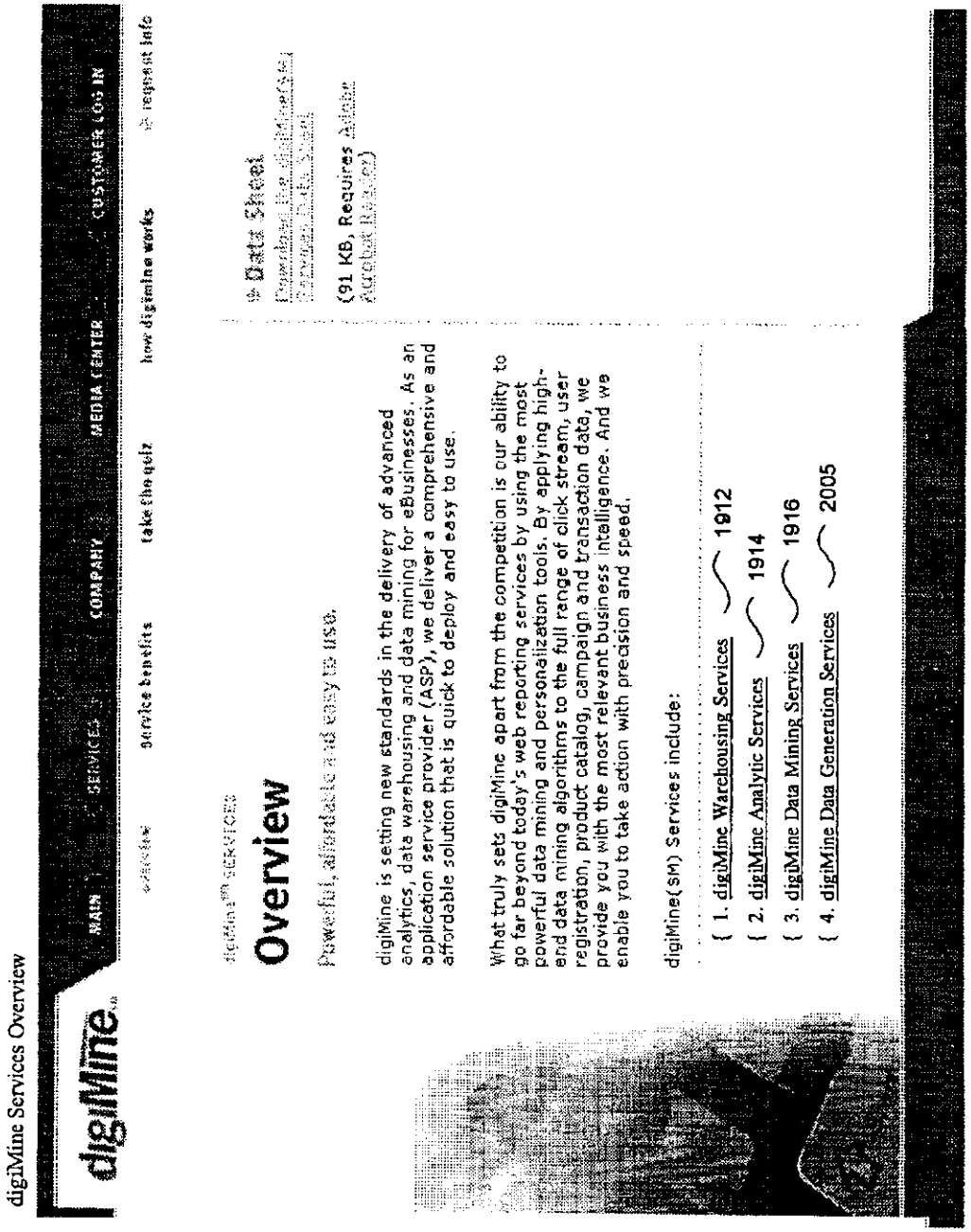
digiMine Services Overview
digiMine
MAIN
SERVICES
COMPANY
MEDIA CENTER
CUSTOMER LOG IN
service benefits
take the quiz
how digimine works
register info
digiMine SERVICES
Overview
Powerful, affordable and easy to use.
digiMine is setting new standards in the delivery of advanced analytics, data warehousing and data mining for eBusinesses. As an application service provider (ASP), we deliver a comprehensive and affordable solution that is quick to deploy and easy to use.
What truly sets digiMine apart from the competition is our ability to go far beyond today's web reporting services by using the most powerful data mining and personalization tools. By applying high-end data mining algorithms to the full range of click stream, user registration, product catalog, campaign and transaction data, we provide you with the most relevant business intelligence. And we enable you to take action with precision and speed.
digiMine(SM) Services include:
1. digiMine Warehousing Services
2. digiMine Analytic Services
3. digiMine Data Mining Services
4. digiMine Data Generation Services
Data Sheet
(91 KB, Requires Adobe Acrobat Reader)
1912
1914
1916
2005
http://www.digimine.com/services/

Fig. 20

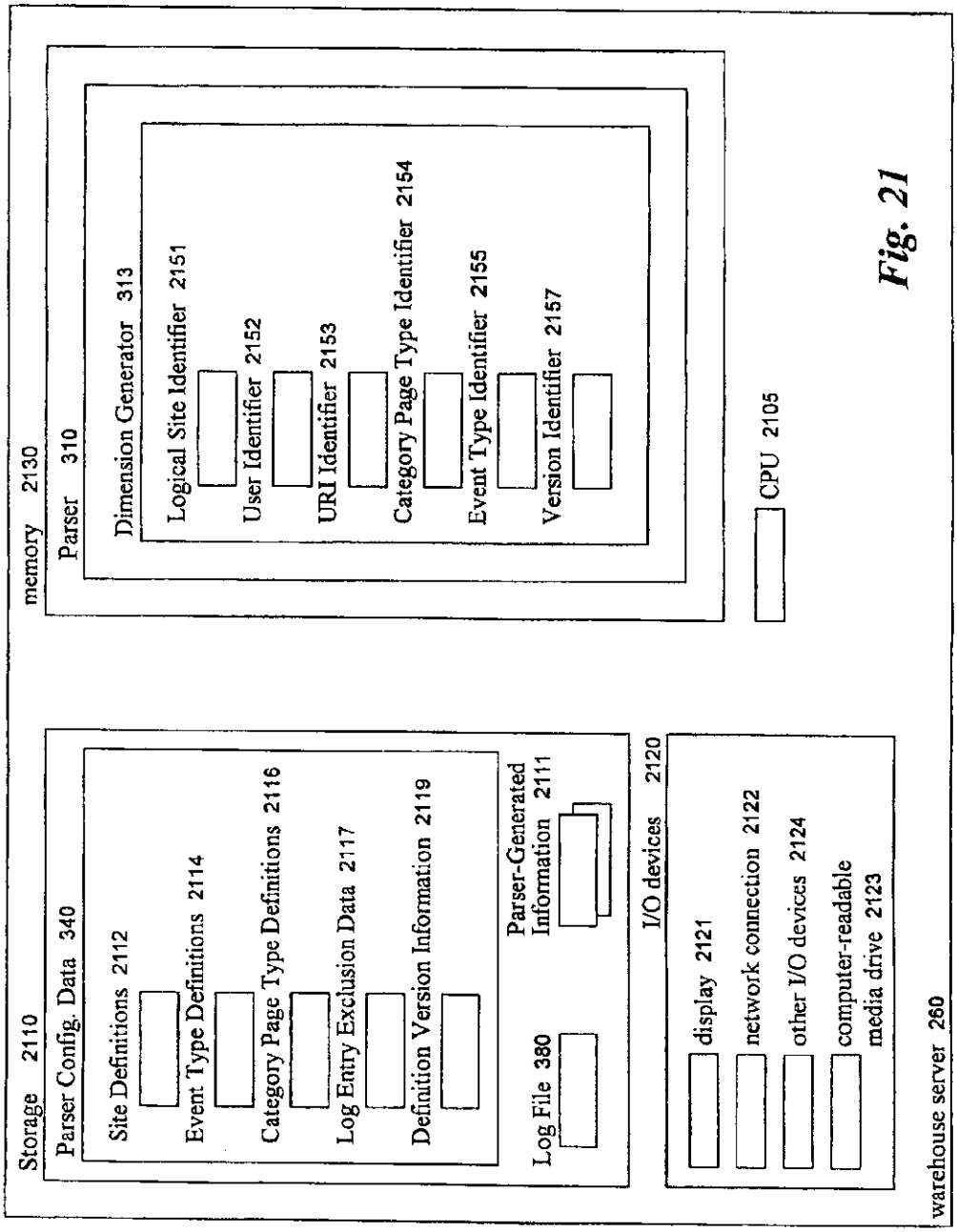
warehouse server 260
Storage 2110
Parser Config. Data 340
Site Definitions 2112
Event Type Definitions 2114
Category Page Type Definitions 2116
Log Entry Exclusion Data 2117
Definition Version Information 2119
Log File 380
Parser-Generated Information 2111
I/O devices 2120
display 2121
network connection 2122
other I/O devices 2124
computer-readable media drive 2123
memory 2130
Parser 310
Dimension Generator 313
Logical Site Identifier 2151
User Identifier 2152
URI Identifier 2153
Category Page Type Identifier 2154
Event Type Identifier 2155
Version Identifier 2157
CPU 2105

Fig. 21

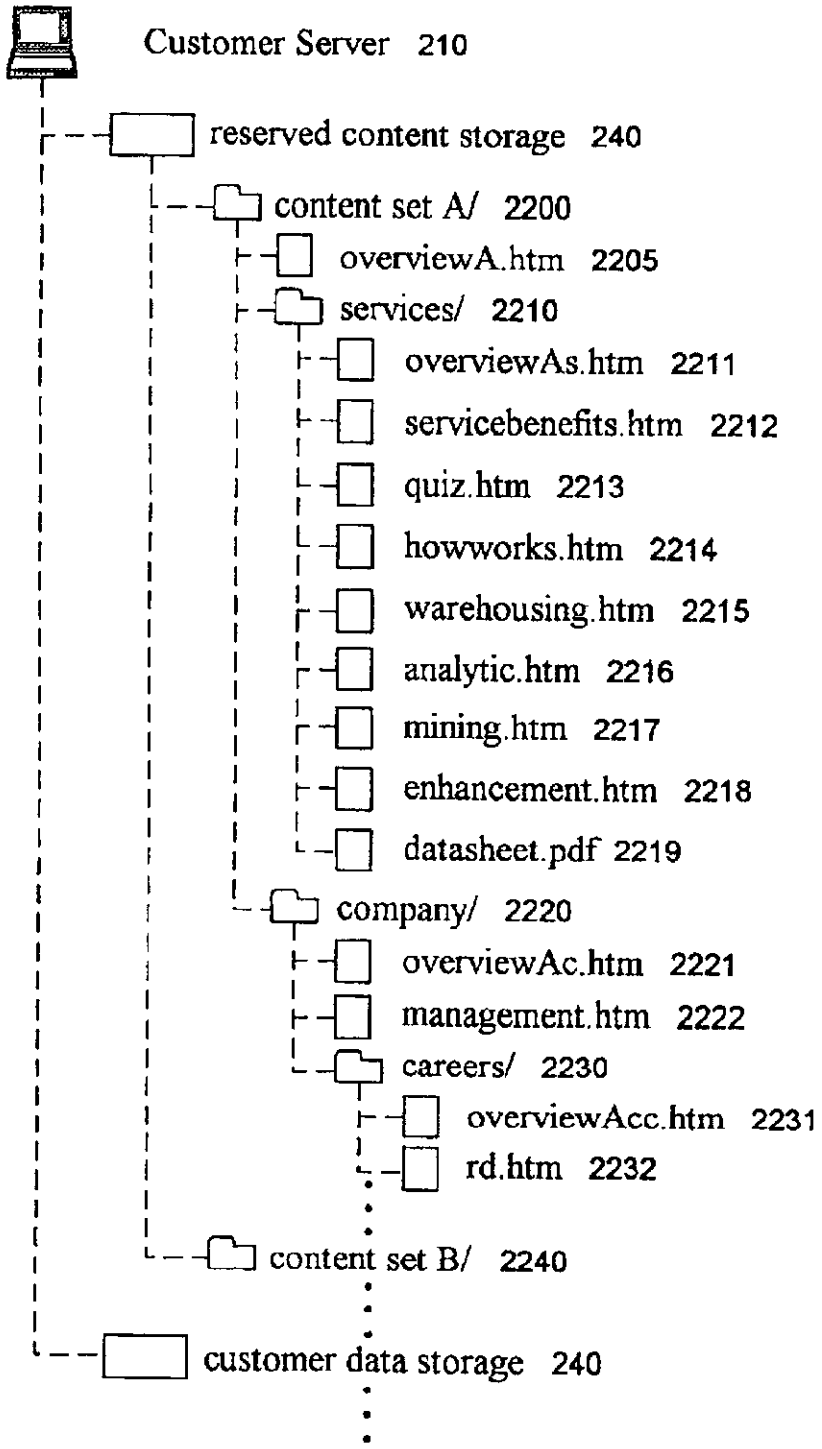
Customer Server 210
reserved content storage 240
content set A/ 2200
overviewA.htm 2205
services/ 2210
overviewAs.htm 2211
servicebenefits.htm 2212
quiz.htm 2213
howworks.htm 2214
warehousing.htm 2215
analytic.htm 2216
mining.htm 2217
enhancement.htm 2218
datasheet.pdf 2219
company/ 2220
overviewAc.htm 2221
management.htm 2222
careers/ 2230
overviewAcc.htm 2231
rd.htm 2232
content set B/ 2240
customer data storage 240

Fig. 22A

Content Set A
Category Hierarchy Table 2250

Category 2251	ID 2252	Category Parent 2253
Services	1	—
Company	2	—
Media Center	3	—
Analysis	4	—
Service Benefits	5	1
Take the Quiz	6	1
⋮		
Careers	20	2
⋮		
R&D	30	20
QA	31	20
⋮		

Content Set A Content Category Table 2260

Content 2261	Category Page Type Definition ID 2262
overviewA.htm	—
overviewAs.htm	1
servicebenefits.htm	1
⋮	
rd.htm	30
⋮	

Fig. 22B

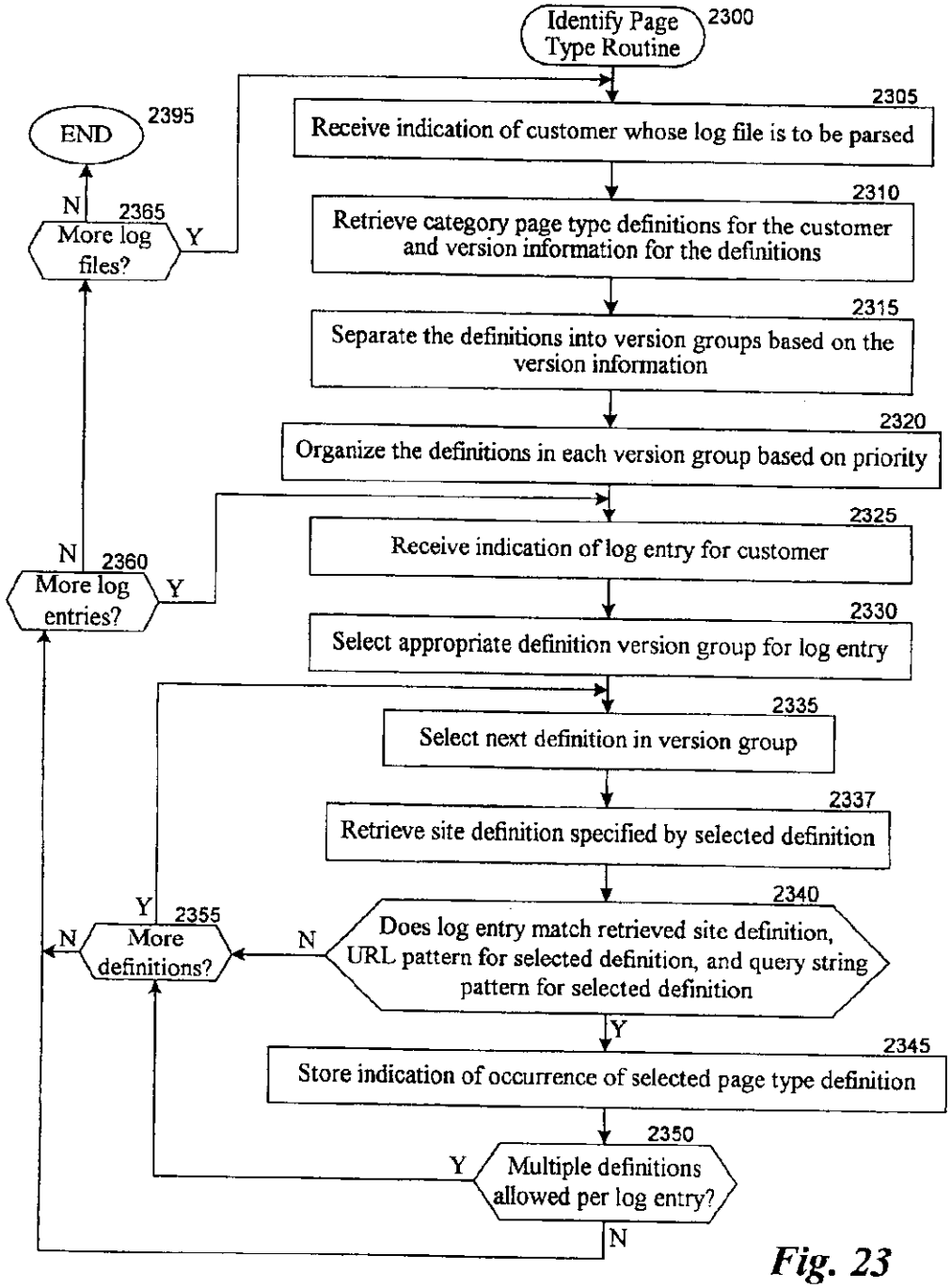
Identify Page Type Routine
2300
2305
Receive indication of customer whose log file is to be parsed
2310
Retrieve category page type definitions for the customer and version information for the definitions
2315
Separate the definitions into version groups based on the version information
2320
Organize the definitions in each version group based on priority
2325
Receive indication of log entry for customer
2330
Select appropriate definition version group for log entry
2335
Select next definition in version group
2337
Retrieve site definition specified by selected definition
2340
Does log entry match retrieved site definition, URL pattern for selected definition, and query string pattern for selected definition
2345
Store indication of occurrence of selected page type definition
2350
Multiple definitions allowed per log entry?
2355
More definitions?
2360
More log entries?
2365
More log files?
2395
END
Y
N

Fig. 23

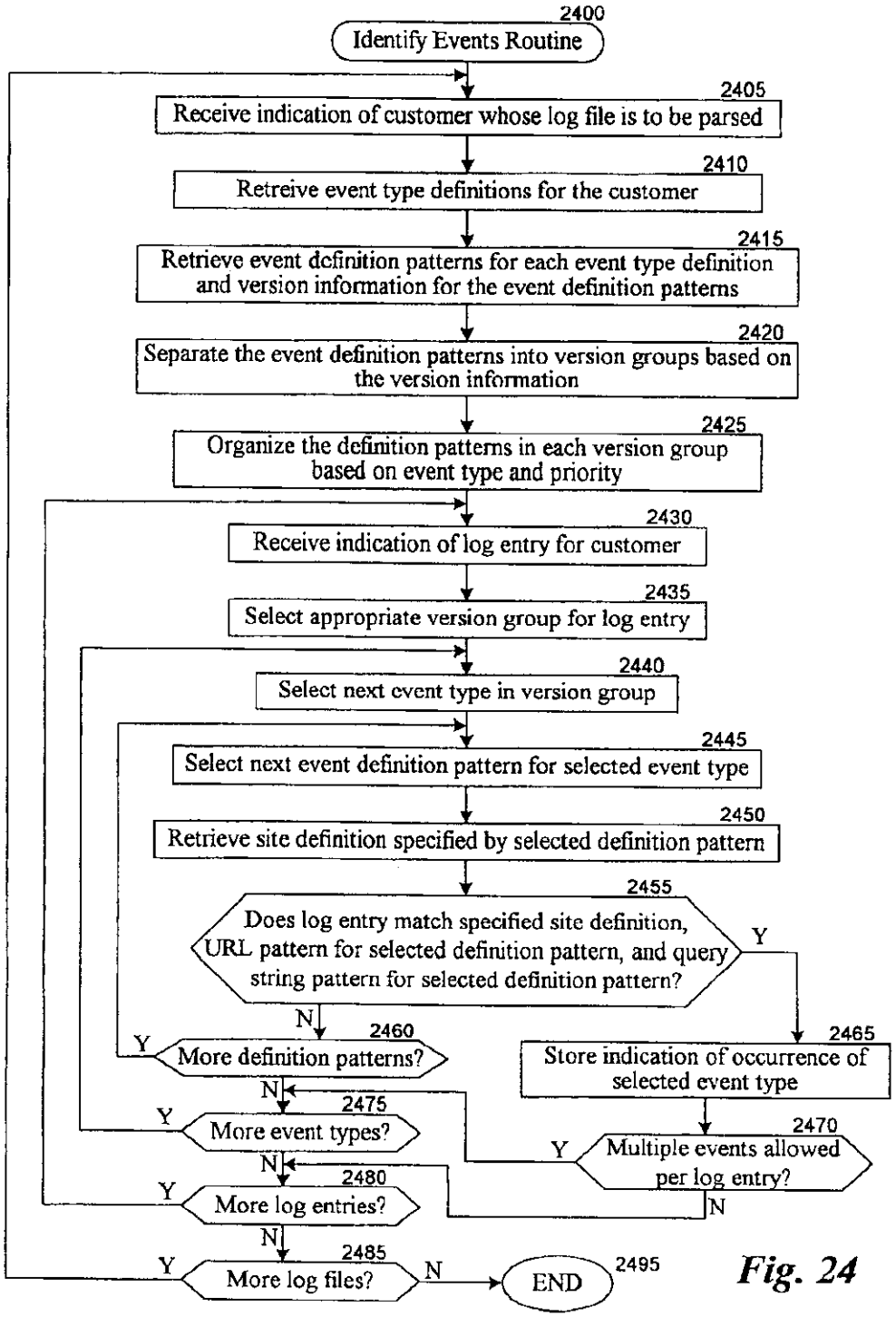
2400
Identify Events Routine
2405
Receive indication of customer whose log file is to be parsed
2410
Retreive event type definitions for the customer
2415
Retrieve event definition patterns for each event type definition and version information for the event definition patterns
2420
Separate the event definition patterns into version groups based on the version information
2425
Organize the definition patterns in each version group based on event type and priority
2430
Receive indication of log entry for customer
2435
Select appropriate version group for log entry
2440
Select next event type in version group
2445
Select next event definition pattern for selected event type
2450
Retrieve site definition specified by selected definition pattern
2455
Does log entry match specified site definition, URL pattern for selected definition pattern, and query string pattern for selected definition pattern?
Y
N
2460
More definition patterns?
Y
N
2465
Store indication of occurrence of selected event type
2470
Multiple events allowed per log entry?
Y
N
2475
More event types?
Y
N
2480
More log entries?
Y
N
2485
More log files?
Y
N
END
2495

Fig. 24

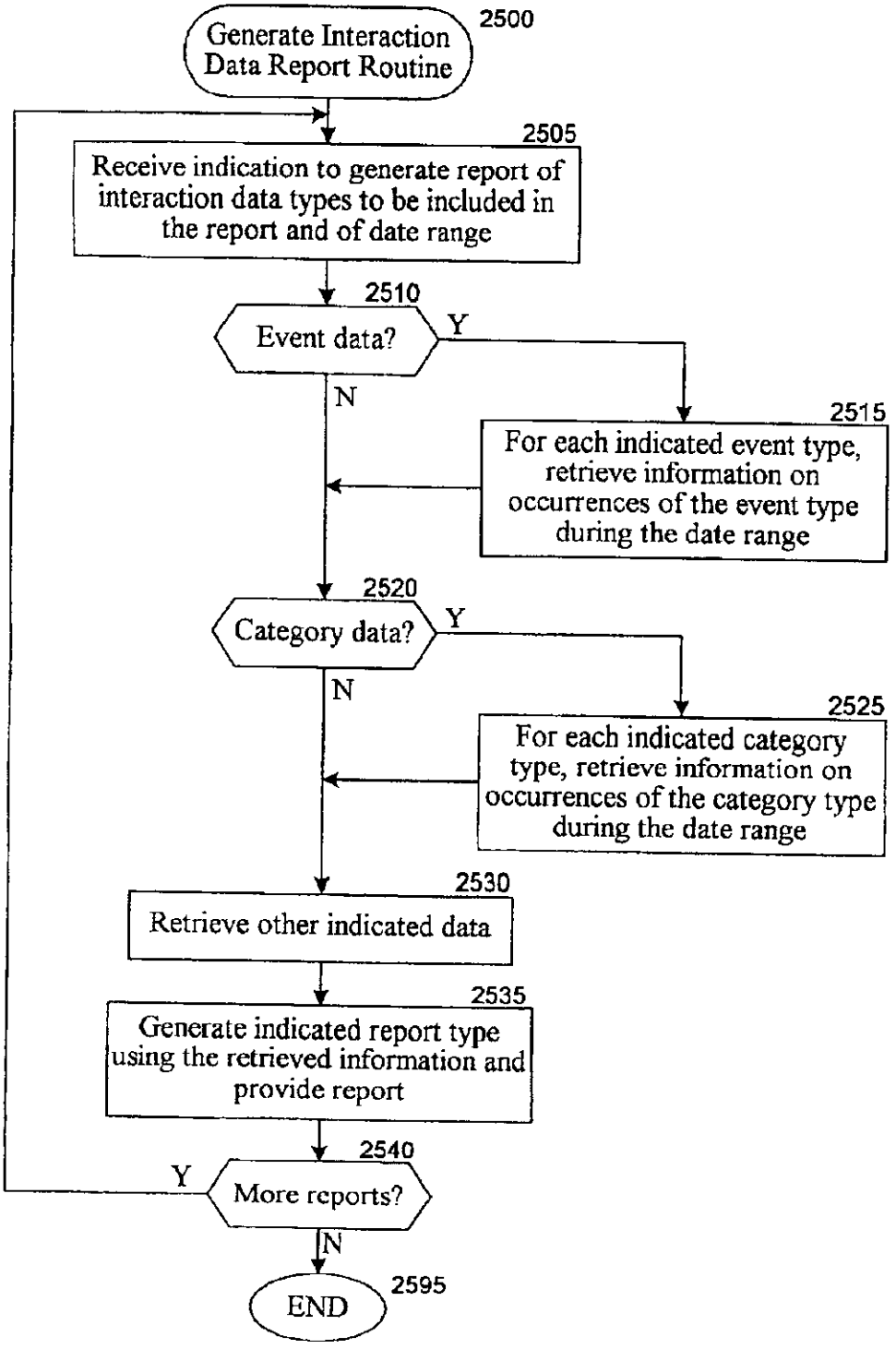
Generate Interaction Data Report Routine
2500
2505
Receive indication to generate report of interaction data types to be included in the report and of date range
2510
Event data?
Y
N
2515
For each indicated event type, retrieve information on occurrences of the event type during the date range
2520
Category data?
Y
N
2525
For each indicated category type, retrieve information on occurrences of the category type during the date range
2530
Retrieve other indicated data
2535
Generate indicated report type using the retrieved information and provide report
2540
More reports?
Y
N
END
2595

Fig. 25

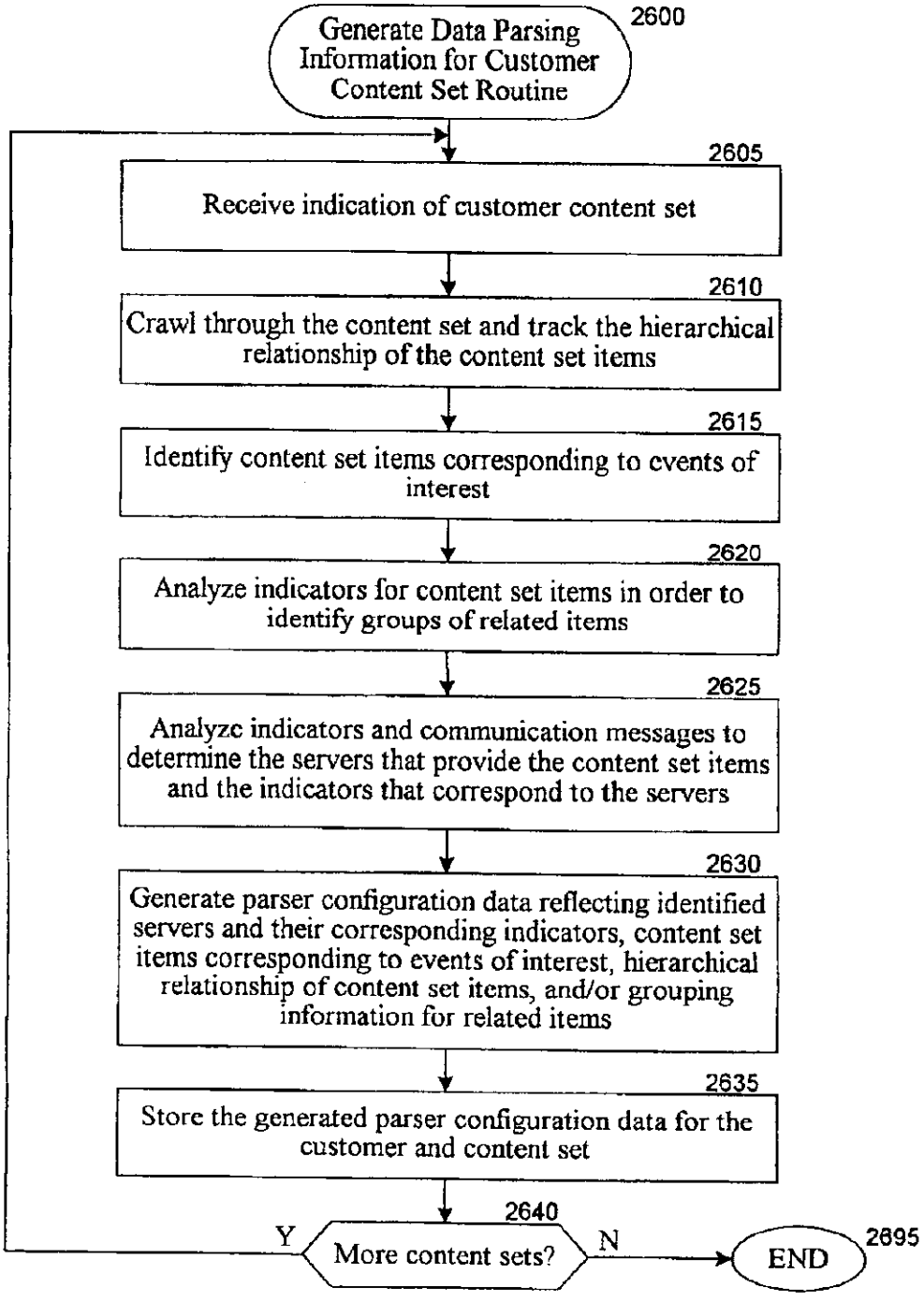
Generate Data Parsing Information for Customer Content Set Routine
2600
2605
Receive indication of customer content set
2610
Crawl through the content set and track the hierarchical relationship of the content set items
2615
Identify content set items corresponding to events of interest
2620
Analyze indicators for content set items in order to identify groups of related items
2625
Analyze indicators and communication messages to determine the servers that provide the content set items and the indicators that correspond to the servers
2630
Generate parser configuration data reflecting identified servers and their corresponding indicators, content set items corresponding to events of interest, hierarchical relationship of content set items, and/or grouping information for related items
2635
Store the generated parser configuration data for the customer and content set
2640
More content sets?
Y
N
END
2695

Fig. 26

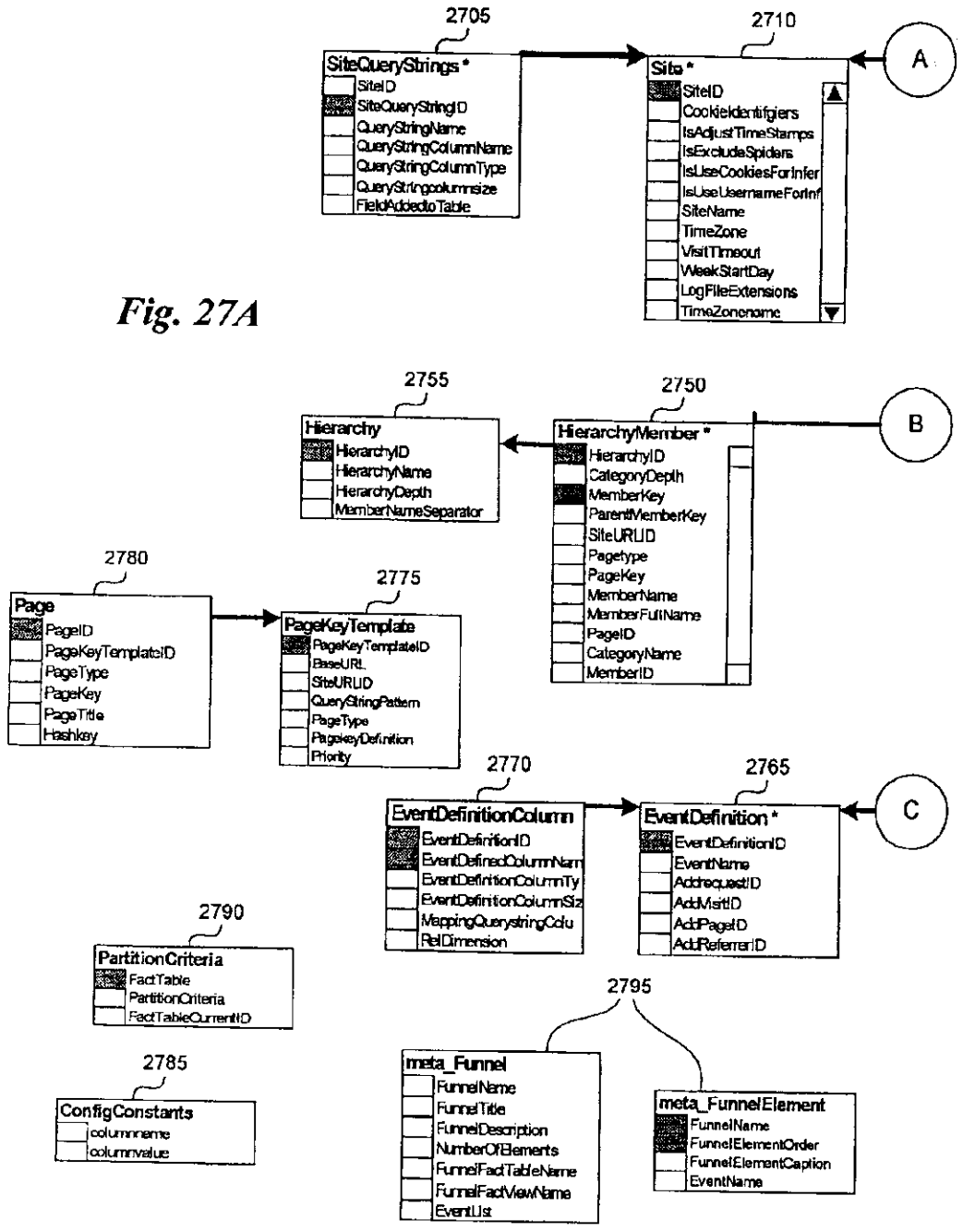
2705
SiteQueryStrings *
SiteID
SiteQueryStringID
QueryStringName
QueryStringColumnName
QueryStringColumnType
QueryStringcolumnsize
FieldAddedtoTable
2710
Site *
SiteID
CookieIdentifiers
IsAdjustTimeStamps
IsExcludeSpiders
IsUseCookiesForInfer
IsUseUsernameForInf
SiteName
TimeZone
VisitTimeout
WeekStartDay
LogFileExtensions
TimeZonename
A
2755
Hierarchy
HierarchyID
HierarchyName
HierarchyDepth
MemberNameSeparator
2750
HierarchyMember *
HierarchyID
CategoryDepth
MemberKey
ParentMemberKey
SiteURLID
Pagetype
PageKey
MemberName
MemberFullName
PageID
CategoryName
MemberID
B
2780
Page
PageID
PageKeyTemplateID
PageType
PageKey
PageTitle
Hashkey
2775
PageKeyTemplate
PageKeyTemplateID
BaseURL
SiteURLID
QueryStringPattern
PageType
PagekeyDefinition
Priority
2770
EventDefinitionColumn
EventDefinitionID
EventDefinedColumnNam
EventDefinitionColumnTy
EventDefinitionColumnSiz
MappingQuerystringColu
RelDimension
2765
EventDefinition *
EventDefinitionID
EventName
AddrequestID
AddVisitID
AddPageID
AddReferrerID
C
2790
PartitionCriteria
FactTable
PartitionCriteria
FactTableCurrentID
2795
meta_Funnel
FunnelName
FunnelTitle
FunnelDescription
NumberOfElements
FunnelFactTableName
FunnelFactViewName
EventList
meta_FunnelElement
FunnelName
FunnelElementOrder
FunnelElementCaption
EventName
2785
ConfigConstants
columnname
columnvalue

Fig. 27A

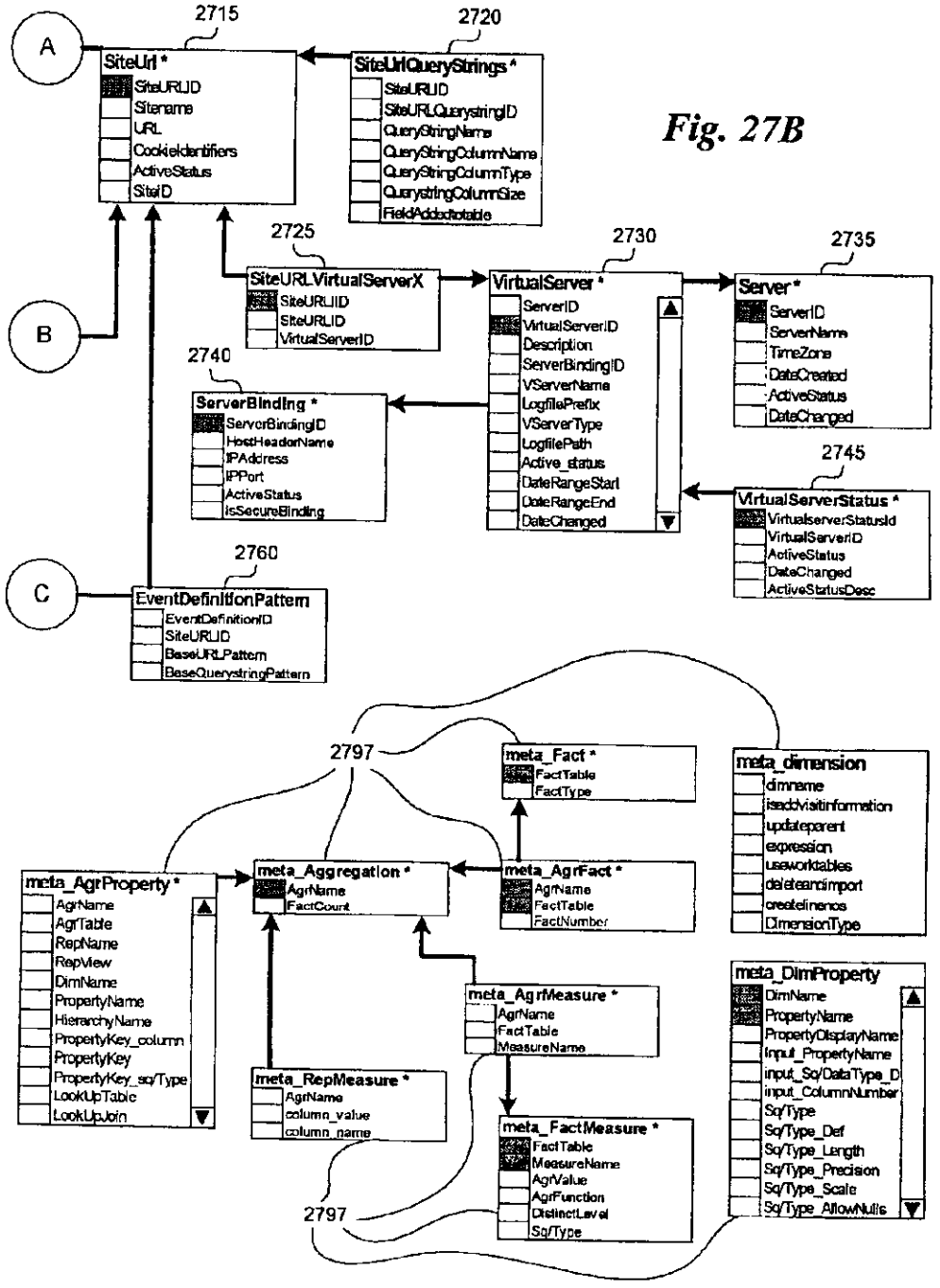
A
B
C
2715
SiteUrl *
SiteURLID
Sitename
URL
CookieIdentifiers
ActiveStatus
SiteID
2720
SiteUrlQueryStrings *
SiteURLID
SiteURLQuerystringID
QueryStringName
QueryStringColumnName
QueryStringColumnType
QuerystringColumnSize
FieldAddedtotable
2725
SiteURLVirtualServerX
SiteURLID
SiteURLID
VirtualServerID
2730
VirtualServer *
ServerID
VirtualServerID
Description
ServerBindingID
VServerName
LogfilePrefix
VServerType
LogfilePath
Active_status
DateRangeStart
DateRangeEnd
DateChanged
2735
Server *
ServerID
ServerName
TimeZone
DateCreated
ActiveStatus
DateChanged
2740
ServerBinding *
ServerBindingID
HostHeaderName
IPAddress
IPPort
ActiveStatus
IsSecureBinding
2745
VirtualServerStatus *
VirtualserverStatusId
VirtualServerID
ActiveStatus
DateChanged
ActiveStatusDesc
2760
EventDefinitionPattern
EventDefinitionID
SiteURLID
BaseURLPattern
BaseQuerystringPattern
2797
meta_Fact *
FactTable
FactType
meta_dimension
dimname
isaddvisitinformation
updateparent
expression
useworktables
deleteamsimport
createlinenos
DimensionType
meta_AgrProperty *
AgrName
AgrTable
RepName
RepView
DimName
PropertyName
HierarchyName
PropertyKey_column
PropertyKey
PropertyKey_sqType
LookUpTable
LookUpJoin
meta_Aggregation *
AgrName
FactCount
meta_AgrFact *
AgrName
FactTable
FactNumber
meta_AgrMeasure *
AgrName
FactTable
MeasureName
meta_RepMeasure *
AgrName
column_value
column_name
meta_FactMeasure *
FactTable
MeasureName
AgrValue
AgrFunction
DistinctLevel
SqType
meta_DimProperty
DimName
PropertyName
PropertyDisplayName
Input_PropertyName
input_SqDataType_D
input_ColumnNumber
SqType
SqType_Def
SqType_Length
SqType_Precision
SqType_Scale
SqType_AllowNulls

Fig. 27B

PARSING NAVIGATION INFORMATION TO IDENTIFY OCCURRENCES CORRESPONDING TO DEFINED CATEGORIES

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of U.S. patent application Ser. No. 09/613,847 filed Jul. 11, 2000, now U.S. Pat. No. 6,785,666 issued Aug. 31, 2004, which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

The described technology relates to analyzing computer interaction or usage data, such as web site navigation information, to identify occurrences corresponding to defined categories.

BACKGROUND

Today's computer networking environments, such as the Internet, offer mechanisms for delivering documents and other information between heterogeneous computer systems. However, in order for a computer to communicate with another computer, the computer must be able to identify and contact that other computer. Computers that are part of the Internet each have a unique numeric identifier, called an "Internet Protocol address," that other computers can use for communication. Thus, when a communication is sent from a client computer to a destination computer over the Internet, the client computer typically specifies the Internet Protocol ("IP") address of the destination computer in order to facilitate the routing of the communication to the destination computer. For example, when a request for a World Wide Web page document ("web page") is sent from a client computer to a web server computer ("web server" or "web site server") from which that web page can be obtained, the client computer typically includes the IP address of the web server.

In order to make the identification of destination computers more mnemonic, a Domain Name System (DNS) is used to translate a unique alphanumeric name for a destination computer, called a "domain name," into the IP address for that computer. For example, the domain name for a hypothetical computer operated by digiMine Corporation ("digiMine") may be "comp23.digimine.com". Using domain names, a user attempting to communicate with this computer could specify a destination of "comp23.digimine.com" rather than the IP address of the computer (e.g., 198.81.209.25).

The subset of Internet sites that comprise the World Wide Web network also supports a standard protocol for requesting and receiving web page documents. This protocol, known as the Hypertext Transfer Protocol (or "HTTP"), defines a message passing protocol for sending and receiving packets of information between diverse applications. Details of HTTP can be found in various documents, including T. Berners-Lee et al., *Hypertext Transfer Protocol—HTTP* 1.0, Request for Comments (RFC) 1945, MIT/LCS, May 1996. Each HTTP message follows a specific layout, which includes among other information, a header which contains information specific to the request or response. Further, each HTTP request message contains a Universal Resource Identifier (or "URI"), which specifies to which network resource the request is to be applied.

Thus, a user can request a particular resource (e.g., a web page or a file) that is available from a web server by

specifying a unique URI for that resource. A URI can be a Uniform Resource Locator ("URL"), Uniform Resource Name ("URN"), or any other formatted string that identifies a network resource. URLs include a protocol to be used in accessing the resource (e.g., "http:" for HTTP), the domain name or IP address of the server providing the resource (e.g., "comp23.digimine.com"), and optionally a server-specific path to the resource (e.g., "/help/HelpPage.html"), thus resulting in the URL "http://comp23.digimine.com/help/HelpPage.html" in this example. In response to a user specifying such a URL, the comp23.digimine.com server would typically return a copy of the "HelpPage.html" file to the user. In addition, in situations where the identified resource corresponds to an executable program on the web server (e.g., a CGI script, Active Server Page (ASP) file, or Java Server Page (JSP) file), the URL can be followed by a query string that will be provided as input to the executable program. Each such query string includes one or more query string parameter names accompanied by a corresponding value (e.g., the parameter names "name1" and "name2" and corresponding values "3" and "ab" in "http://www.digimine.com/search.asp?name1=3&name2=ab"). URLs are discussed in detail in T. Berners-Lee, et al., *Uniform Resource Locators (URL)*, RFC 1738, CERN, Xerox PARC, Univ. of Minn., December 1994.

FIG. **1** illustrates how a browser application enables users to navigate among nodes on the web network by requesting and receiving web pages. For the purposes of this discussion, a web page is any type of document that abides by the HTML format. That is, the document includes an "<HTML>" statement. Thus, a web page is also referred to as an HTML document. The HTML format is a document mark-up language, defined by the Hypertext Markup Language ("HTML") specification. HTML defines tags for specifying how to interpret the text and images stored in an HTML document. For example, there are HTML tags for defining paragraph formats and for emboldening and underlining text. In addition, the HTML format defines tags for adding images to documents and for formatting and aligning text with respect to images. HTML tags appear between angle brackets, for example, <HTML>. Further details of HTML are discussed in T. Berners-Lee and D. Connolly, *Hypertext Markup Language*-2.0, RFC 1866, MIT/W3C, November 1995.

In FIG. **1**, a web browser application **101** is shown executing on a client computer **102**, which communicates with a server computer **103** by sending and receiving HTTP packets (messages). HTTP messages may also be generated by other types of computer programs, such as spiders and crawlers. The web browser "navigates" to new locations on the network to browse (display) what is available at these locations. In particular, when the web browser "navigates" to a new location, it requests a new document from the new location (e.g., the server computer) by sending an HTTP-request message **104** using any well-known underlying communications wire protocol. The HTTP-request message follows the specific layout discussed above, which includes a header **105** and a URI field **106**, which specifies the network location to which to apply the request. When the server computer specified by URI receives the HTTP-request message, it interprets the message packet and sends a return message packet to the source location that originated the message in the form of an HTTP-response message **107**. It also stores a copy of the request and basic information about the requesting computer in a log file. In addition to the standard features of an HTTP message, such as the header **108**, the HTTP-response message contains the

requested HTML document **109**. When the HTTP-response message reaches the client computer, the web browser application extracts the HTML document from the message, and parses and interprets (executes) the HTML code in the document and displays the document on a display screen of the client computer as specified by the HTML tags. HTTP can also be used to transfer other media types, such as the Extensible Markup Language ("XML") and graphics interchange format ("GIF") formats.

The World Wide Web is especially conducive to conducting electronic commerce ("e-commerce"). E-commerce generally refers to commercial transactions that are at least partially conducted using the World Wide Web. For example, numerous web sites are available through which a user using a web browser can purchase items, such as books, groceries, and software. A user of these web sites can browse through an electronic catalog of available items to select the items to be purchased. To purchase the items, a user typically adds the items to an electronic shopping cart and then electronically pays for the items that are in the shopping cart. The purchased items can then be delivered to the user via conventional distribution channels (e.g., an overnight courier) or via electronic delivery when, for example, software is being purchased. Many web sites are also informational in nature, rather than commercial in nature. For example, many standards organizations and governmental organizations have web sites with a primary purpose of distributing information. Also, some web sites (e.g., a search engine) provide information and derive revenue from advertisements that are displayed.

The success of any web-based business depends in large part on the number of users who visit the business's web site and that number depends in large part on the usefulness and ease-of-use of the web site. Web sites typically collect extensive information on how its users use the site's web pages. This information may include a complete history of each HTTP request received by and each HTTP response sent by the web site. The web site may store this information in a navigation file, also referred to as a log file or click stream file. By analyzing this navigation information, a web site operator may be able to identify trends in the access of the web pages and modify the web site to make it easier to use and more useful. Because the information is presented as a series of events that are not sorted in a useful way, many software tools are available to assist in this analysis. A web site operator would typically purchase such a tool and install it on one of the computers of the web site. There are several drawbacks with the use of such an approach of analyzing navigation information. First, the analysis often is given a low priority because the programmers are typically busy with the high priority task of maintaining the web site. Second, the tools that are available provide little more than standard reports relating to low-level navigation through a web site. Such reports are not very useful in helping a web site operator to visualize and discover high-level access trends. Recognition of these high-level access trends can help a web site operator to design the web site. Third, web sites are typically resource intensive, that is they use a lot of computing resources and may not have available resources to effectively analyze the navigation information.

It would also be useful to analyze the execution of computer programs other than web server programs. In particular, many types of computer programs generate events that are logged by the computer programs themselves or by other programs that receive the events. If a computer program does not generate explicit events, another program may be able to monitor the execution and generate events on behalf of that computer program. Regardless of how event data is collected, it may be important to analyze that data. For example, the developer of an operating system may want to track and analyze how the operating system is used so that the developer can focus resources on problems that are detected, optimize services that are frequently accessed, and so on. The operating system may generate a log file that contains entries for various types of events (e.g., invocation of a certain system call).

Thus, as noted above, interaction or usage data (e.g., web site navigation information or computer program event information) can contain important low-level information about interactions and usage that have occurred, but current techniques for extracting high-level summaries or analyzing such interactions or usage are limited. For example, it would be useful in many situations to know the number of occurrences of interactions or uses of a specified category or type during a specified time period, or to know how such occurrences relate to other occurrences of interest. Similarly, when a sequence of interactions or uses is of interest, it would be useful to know the number of occurrences of each interaction or usage in the sequence. In addition, analysis of interaction or usage data is further complicated when the format or content types of such data changes over time, such as to reflect changes in a corresponding web site or computer program. It would therefore be useful to have techniques for effectively identifying and extracting useful high-level information from interaction or usage data, and for tracking changes in the format or content type of the interaction or usage data. Accordingly, techniques for analyzing interaction and usage data to obtain such information would have significant utility.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates how a browser application enables users to navigate among nodes on the web network by requesting and receiving web pages.

FIG. **2A** is a block diagram illustrating components of the data warehouse system in one embodiment,

FIG. **2B** is a block diagram illustrating details of the components of the data warehouse system in one embodiment.

FIG. **3** is a block diagram illustrating the sub-components of the data processor component in one embodiment.

FIG. **4** is a block diagram illustrating some of the tables of the local data warehouse and the main data warehouse in one embodiment.

FIG. **5** is a flow diagram illustrating the parse log data routine that implements the parser in one embodiment.

FIG. **6** is a flow diagram of the filter log entry routine in one embodiment.

FIG. **7** is a flow diagram illustrating the normalize log entry routine.

FIG. **8** is a flow diagram of the generate dimensions routine in one embodiment.

FIG. **9** is a flow diagram of the identify logical site routine in one embodiment.

FIG. **10** is a flow diagram of the identify user routine in one embodiment.

FIG. **11** is a flow diagram of the identify page type routine in one embodiment.

FIG. **12** is a flow diagram illustrating the identify events routine in one embodiment.

FIG. **13** is a flow diagram illustrating the identify sessions routine in one embodiment.

FIG. **14** is a flow diagram of the generate aggregate statistics routine in one embodiment.

FIG. **15** is a flow diagram of the import log data routine implementing the importer in one embodiment.

FIG. **16** is a flow diagram of the load dimension table routine and one embodiment.

FIG. **17** is a flow diagram of the load fact table routine in one embodiment.

FIG. **18** is a flow diagram illustrating the identify user aliases routine in one embodiment.

FIGS. **19A–19AE** illustrate example customer web pages for which parser configuration data can be specified.

FIG. **20** illustrates an example updated version of a customer web page.

FIG. **21** is a block diagram illustrating details of the components of the data warehouse server in one embodiment.

FIGS. **22A** and **22B** illustrate example hierarchical category information for customer web pages.

FIG. **23** is a flow diagram illustrating an embodiment of the Identify Page Type routine.

FIG. **24** is a flow diagram illustrating an embodiment of the Identify Events routine.

FIG. **25** is a flow diagram illustrating an embodiment of the Generate Interaction Data Report routine.

FIG. **26** is a flow diagram illustrating an embodiment of the Generate Data Parsing Information For Customer Content Set routine.

FIGS. **27A** and **27B** illustrate an example of data structures used to store parser configuration data.

DETAILED DESCRIPTION

A method and system for providing customers with access to and analysis of interaction or usage data (e.g., navigation data collected at customer web sites or computer program event information) is provided. The interaction or usage data, hereinafter "interaction data" or "event data," may be stored in log files and supplemented with data from other sources, such as product databases and customer invoices. In one embodiment, a data warehouse system collects customer data from the customer web sites and stores the data at a data warehouse server. The customer data may include application event data (e.g., click stream log files), user attribute data of users of the customer web site (e.g., name, age, and gender), product data (e.g., catalog of products offered for sale by the customer), shopping cart data (i.e., identification of the products currently in a user's shopping cart), and so on. The data warehouse server interacts with the customer servers to collect the customer data on a periodic basis. The data warehouse server may provide instructions to the customer servers identifying the customer data that is to be uploaded to the data warehouse server. These instructions may include the names of the files that contains the customer data and the name of the web servers on which the files reside. These instructions may also indicate the time of the day when the customer data is to be uploaded to the data warehouse server.

When the data warehouse server receives customer data, it converts the customer data into a format that is more conducive to processing by decision support system applications used to analyze customer data. For example, the data warehouse server may analyze low-level navigation events (e.g., each HTTP request that is received by the customer web site) to identify high-level events (e.g., a user session). The data warehouse server then stores the converted data into a data warehouse. The data warehouse server functions as an application service provider that provides various decision support system applications for the customers. For example, the data warehouse server provides decision support system applications to analyze and graphically display the results of the analysis for a customer. The decision support system applications may be accessed through a web browser. In one embodiment, the customer servers are connected to the data warehouse server via the Internet and the data warehouse server provides data warehousing services to multiple customers.

The data warehouse system may provide a data processor component that converts the log files into a format that is more conducive to processing by the decision support system applications. In one embodiment, the converted data is stored in a data warehouse that includes fact and dimension tables. Each fact table contains entries corresponding to a type of fact derived from the log files. For example, a web page access fact table may contain an entry for each web page access identified in the log files. Each entry may reference attributes of the web page access, such as the identity of the web page and identity of the accessing user. The values for each attribute are stored in a dimension table for that attribute. For example, a user dimension table may include an entry for each user and the entries of the web access fact table may include a user field that contains an index (or some other reference) to the entry of the user dimension table for the accessing user. The user dimension table may contain the names of the users and other user-specific information. Alternatively, the user dimension table may itself also be a fact table that includes references to dimension tables for the attributes of users. The data warehouse may also include fact tables and dimension tables that represent high-level facts and attributes derived from the low-level facts and attributes of the log files. For example, high-level facts and attributes may not be derivable from only the data in a single log entry. For example, the higher level category (e.g., shoes or shirts) of a web page may be identified using a mapping of web page URIs to categories. These categories may be stored in a category dimension table. Also, certain facts, such as the collection of log entries that comprise a single user web access session or visit, may only be derivable by analyzing a series of log entries.

The data processor component may have a parser component and a loader component. The parser of the data processor parses and analyzes a log file and stores the resulting data in a local data warehouse that contains information for only that log file. The local data warehouse may be similar in structure (e.g., similar fact and dimension tables) to the main data warehouse used by decision support system applications. The local data warehouse may be adapted to allow efficient processing by the parser. For example, the local data warehouse may be stored in primary storage (e.g., main memory) for speed of access, rather than in secondary storage (e.g., disks). The parser may use parser configuration data that defines, on a customer-by-customer basis, the high-level data to be derived from the log entries. For example, the parser configuration data may specify the mapping of URIs to web page categories. The loader of the data processor transfers the data from the local data warehouse to the main data warehouse. The loader may create separate partitions for the main data warehouse. These separate partitions may hold the customer data for a certain time period (e.g., a month's worth of data). The loader adds entries to the main fact tables (i.e., fact tables of the main data warehouse) for each fact in a local fact table (i.e., fact

table of the local data warehouse). The loader also adds new entries to the main dimension tables to represent attribute values of the local dimension tables that are not already in the main dimension tables. The loader also maps the local indices (or other references) of the local dimension tables to the main indices used by the main dimension tables.

FIG. **2A** is a block diagram illustrating components of the data warehouse system in one embodiment. The data warehouse system includes customer components that execute on the customer servers and data warehouse components that execute on the data warehouse server. The customer servers **210** and the data warehouse server **260** are interconnected via the Internet **250**. Customer components executing on a customer server includes a data collection component **220** and a data viewer **230**. The data viewer may reside on a client computer of the customer, rather than a server. The data collection component collects the customer data from the storage devices **240** of the customer servers. The data viewer provides access for viewing of data generated by the decision support system applications of the data warehouse server. In one embodiment, the data viewer may be a web browser. The data warehouse server includes a data receiver component **270**, the data processor component **280**, the data warehouse **290**, and decision support system applications **291**. The data receiver component receives customer data sent by the data collection components executing at the various customer web sites. The data processor component processes the customer data and stores it in the data warehouse. The decision support system application provides the customer with tools for analyzing and reviewing the customer data that is stored in the main data warehouse. Analysis performed on and report generated from customer data are described in U.S. patent application Ser. No. 09/638,836, entitled "Identifying And Reporting On Combinations Of Events In Usage Data" and filed Aug. 14, 2000, now abandoned; U.S. patent application Ser. No. 09/742,685, entitled "Report Depicting Extent Of Completion Of A Process" and filed Dec. 20, 2000, now abandoned; and U.S. patent application Ser. No. 09/613,846, entitled "Web-Based Extraction And Display Of Information For Graphical Structures" and filed Jul. 11, 2000, now abandoned, each of which are hereby incorporated by reference. In one embodiment, each customer has its own dimension and fact tables so that multiple customers' information is not intermingled.

FIG. **2B** is a block diagram illustrating details of the components of the data warehouse system in one embodiment. The data collection component **220** includes a monitor sub-component **221** and a pitcher sub-component **222**. The data collection component is described in more detail in U.S. patent application Ser. No. 09/613,845, entitled "Method and System for Monitoring a Resource via the Web" and filed Jul. 11, 2000, now abandon which is hereby incorporated by reference. The pitcher is responsible for retrieving instructions from the data warehouse server, collecting the customer data in accordance with the retrieved instructions, and uploading the customer data to the data warehouse server. The monitor is responsible for monitoring the operation of the pitcher and detecting when the pitcher may have problems in collecting and uploading the customer data. When the monitor detects that a problem may occur, it notifies the data warehouse server so that corrective action may be taken in advance of the collecting and uploading of the customer data. For example, the pitcher may use certain log on information (e.g., user ID and password) to access a customer web server that contains customer data to be uploaded. The monitor may use that log on information to verify that the log on information will permit access to the customer data. Access may be denied if, for example, a customer administrator inadvertently deleted from the customer web server the user ID used by the pitcher. When the monitor provides advance notification of a problem, the problem might be corrected before the pitcher attempts to access the customer data. The monitor also periodically checks the pitcher to ensure that the pitcher is executing and, if executing, executing correctly.

The data receiver component of the data warehouse server includes a status receiver sub-component **271**, a catcher sub-component **272**, an FTP server **273**, a status database **274**, and a collected data database **275**. The status receiver receives status reports from the customer servers and stores the status information in the status database. The catcher receives and processes the customer data that is uploaded from the customer web sites and stores the data in the collected data database.

The data processor component includes a parser sub-component **281** and a loader sub-component **282**. The parser analyzes the low-level events of the customer data and identifies high-level events and converts the customer data into a format that facilitates processing by the decision support system applications. The loader is responsible for storing the identified high-level events in the data warehouse **290**. In one embodiment, a customer may decide not to have the data collection component executing on its computer systems. In such a case, the customer server may include an FTP client **245** that is responsible for periodically transferring the customer data to the FTP server **273** of the data warehouse server. The data receiver may process this customer data at the data warehouse server in the same way as the pitcher processes the data at the customer servers. The processed data is then stored in the collected data database.

FIG. **3** is a block diagram illustrating the sub-components of the data processor component in one embodiment. The data processor component **300** includes a parser **310**, data storage area **320**, and a loader **330**. The data processor component inputs parser configuration data **340** and a log file **350** and updates the main data warehouse **360**. The parser configuration data may include a mapping of actual web sites to logical sites and a mapping of a combination of Uniform Resource Identifiers ("URIs") and query strings of the log entries to page definitions (e.g., categories) and event definitions. The parser processes the entries of the log file to generate facts and dimensions to eventually be stored in the main data warehouse. The parser identifies events in accordance with the parser configuration data. The parser includes a filter log entry component **311**, a normalize log entry component **312**, a generate dimensions component **313**, an identify sessions component **314**, and a generate aggregate statistics component **315**. The filter log entry component identifies which log entries should not be included in the main data warehouse. For example, a log entry that has an invalid format should not be included. The normalize log entry component normalizes the data in a log entry. For example, the component may convert all times to Greenwich Mean Time ("GMT"). The generate dimensions component identifies the various dimensions related to a log entry. For example, a dimension may be the Uniform Resource Identifier of the entry or the logical site identifier. The identify sessions component processes the parsed log file data stored in the local data warehouse to identify user sessions. A user session generally refers to the concept of a series of web page accesses that may be related in some way, such as by temporal proximity. The generate aggregate statistics component aggregates data for the log file being processed as each log entry is processed or after the log file is parsed. The

data storage area **320** includes a local data warehouse **321**. In one embodiment, the local data warehouse is stored non-persistently (or temporarily) in main memory of the computer system. The local data warehouse may contain fact tables and dimension tables that correspond generally to the tables of the main data warehouse **360**. The loader retrieves the information from the local data warehouse and stores the information in the main data warehouse. The loader includes a create partitions component **331**, a load dimension table component **332**, and a load fact table component **333**. The create partitions components creates new partitions for the main data warehouse. A partition may correspond to a collection of information within a certain time range. For example, the main data warehouse may have a partition for each month, which contains all the data for that month. The load dimension table component and the load fact table component are responsible for loading the main data warehouse with the dimensions and facts that are stored in the local data warehouse.

In one embodiment, the log file is a web server log file of a customer. The log file may be in the "Extended Log File Format" as described in the document "http://www.w3.org/TR/WD-logfile-960323" provided by the World Wide Web Consortium, which is hereby incorporated by reference. According to that description, the log file contains lines that are either directives or entries. An entry corresponds to a single HTTP transaction (e.g., HTTP request and an HTTP response) and consists of a sequence of fields (e.g., integer, fixed, URI, date, time, and string). The meaning of the fields in an entry is specified by a field directive specified in the log file. For example, a field directive may specify that a log entry contains the fields date, time, client IP address, server IP address, and success code. Each entry in the log file would contain these five fields.

The parser configuration data defines logical sites, page definitions, and event definitions. A logical site is a collection of one or more IP addresses and ports that should be treated as a single web site. For example, a web site may actually have five web servers with different IP addresses that handle HTTP requests for the same domain. These five IP addresses may be mapped to the same logical site to be treated as a single web site. The page definitions define the format of the URIs of log entries that are certain page types. For example, a URI with a query string of "category=shoes" may indicate a page type of "shoes." Each event definition defines an event type and a value for that event type. For example, a log entry with a query string that includes "search=shoes" represents an event type of "search" with an event value of "shoes." Another log entry with a query string of "add=99ABC" may represent an event type of "add" an item to the shopping cart with an event value of item number "99ABC."

FIG. **4** is a block diagram illustrating some of the tables of the local data warehouse and the main data warehouse in one embodiment. These data warehouses are databases that include fact tables and dimension tables. A fact table contains an entry for each instance of fact (e.g., web page access). A dimension table contains an entry for each possible attribute value of an attribute (e.g., user). The entries of a fact table contain dimension fields that refer to the entries into the dimension tables for their attribute values. A table may be both a fact table and a dimension table. For example, a user dimension table with an entry for each unique user may also be a fact table that refers to attributes of the users that are stored in other dimension tables. The data warehouses contain a log entry table **401**, a user table **402**, a logical site table **403**, a URI table **404**, a referrer URI table **405**, a page type table **406**, event type tables **407**, a query string table **408**, and a referrer query string table **409**. The log entry table is a fact table that contains an entry for each log entry that is not filtered out by the parser. The other tables are dimension tables for the log entry table. The user table contains an entry for each unique user identified by the parser. The logical site table contains an entry for each logical site as defined in the parser configuration data. The URI table contains an entry for each unique URI of an entry in the log entry table. The referrer URI table contains an entry for each referrer URI of the log entry table. The page type table contains an entry for each page type identified by the parser as defined in the parser configuration data. The data warehouse contains an event table for each type of event defined in the parser configuration data. Each event table contains an entry for each event value of that event type specified in an entry of the log entry table. The query string table contains an entry for each unique query string identified in an entry of the log entry table. The referrer query string contains an entry for each unique referrer query string identified in an entry of the log entry table.

Table 1 is an example portion of a log file. The "#fields" directive specifies the meaning of the fields in the log entries. Each field in a log entry is separated by a space and an empty field is represented by a hyphen. The #fields directive in this example indicates that each entry includes the date and time when the transaction was completed (i.e., "date" and "time"), the client IP address (i.e., "c-ip"), and so on. For example, the first log entry has a data and time of "2000-06-01 07:00:04" and a client IP address of "165.21.83.161."

TABLE 1

#Software: Microsoft Internet Information Server 4 0
#Version 1 0
#Date: 2000-06-01 07 00:04
#Fields date time c-ip cs-username s-sitename s-computername s-ip cs-method cs-uri-stem cs-uri-query sc-status sc-win32-status sc-bytes cs-bytes time-taken s-port cs-version cs(User-Agent) cs(Cookie) cs(Referrer)
2000-06-01 07:00:04 165 21 83 161 - W3SVC2 COOK_002 206.191.163 41 GET /directory/28.ASP - 200 0 148428 369 9714 80 HTTP/1.0 Mozilla/3.04+(Win95,+I) ASPSESSIONIDQQGGQGPG=JBCCFIPBBHHDANBAFFIGLGPH http.//ecommerce com/Default asp
2000-06-01 07:00:20 4.20.197 70 - W3SVC2 COOK_002 206 191.163 41 GET /Default.asp - 302 0 408 259 30 80 HTTP/1.0 Mozilla/4.0+(compatible;+Keynote-Perspective+4.0) --
2000-06-01 07:00:20 4.20.197.70 - W3SVC2 COOK_002 206 191.163 41 GET /Default.asp - 200 0 41245 266 200 80 HTTP/1.0 Mozilla/4.0+(compatible,+Keynote-Perspective+4.0) --
2000-06-01 07:00:27 204.182.65.192 - W3SVC2 COOK_002 206.191.163.41 HEAD /Default.asp - 302 0 254 66 40 80 HTTP/1 0 Ipswitch_ WhatsUp/3.0 --
2000-06-01 07:00 32 24 10 69 137 - W3SVC2 COOK_002 206.191.163.41 GET /directory/541.asp - 200 0 22427 459 421 80 HTTP/1.0 Mozilla/4.7+[en]+(Win98,+U)

TABLE 1-continued

```
ASPSESSIONIDQQGGQGPG=BHBCFIPBEJPNOMDPKCGLKNGC;
+ARSiteUser=1%2DC2B25364%2D3775%2D11D4%2DBAC1%2D0050049BD2E4,+ARSites=ALR=1
http://ecommerce.com/directory/34.asp
2000-06-01 07:00:34 192.102.216 101 - W3SVC2 COOK_002 206 191 163.41 GET /encyc/terms/L/7276 asp -
200 0 20385 471 290 80 HTTP/1.0 Mozilla/4 7+[en]+(X11.+I,+SunOS+5 5 1+sun4u)
ASPSESSIONIDQQGGQGPG=PKBCFIPBIKONBPDHKDMMEHCE
http://search ecommerce.com/gsearchresults.asp?site=ecommerce&ecommerce=ecommerce& allsites=1&q1=loin
2000-06-01 07:00:34 216.88.216.227 - W3SVC2 COOK_002 206.191.163.41 GET /default.asp - 200 0 41253 258
180 80 HTTP/1.1 Mozilla/4.0+(compatible:+MSIE+4.01,+MSN+2.5;+MSN+2.5;+Windows+98) --
2000-06-01 07:00:36 199.203.4.10 - W3SVC2 COOK_002 206.191.163.41 GET /Default.asp - 302 0 408 485 30
80 HTTP/1 0 Mozilla/4.0+(compatible,+MSIE+5 01,+Windows+98;+TUCOWS)
SITESERVER=ID=22f117fb3708b2278f3c 426796a78e2a -
2000-06-01 07:00:37 199 203.4 10 - W3SVC2 COOK_002 206.191.163 41 GET /Default.asp - 200 0 41277 492
421 80 HTTP/1 0 Mozilla/4.0+(compatible,+MSIE+5 01,+Windows+98,+TUCOWS)
SITESERVER=ID=22f117fb3708b2278f3c 426796a78e2a -
2000-06-01 07:00:43 24.10 69 137 - W3SVC2 COOK_002 206.191.163.41 GET /directory/34.asp - 200 0 17835
458 320 80 HTTP/1.0
Mozilla/4.7+[en]+(Win98;+U)ASPSESSIONIDQQGGQGPG=BHBCFIPBEJPNOMDPKCGLKNGC;
+ARSiteUser=1%2DC2B25364%2D3775%2D11D4%2DBAC1%2D0050049BD2E4;+ARSites=ALR=1
http.//ecommerce.com/directory/25.asp
2000-06-01 07:00:47 199.203 4.10 - W3SVC2 COOK_002 206 191 163 41 GET /jumpsite asp
jumpsite=5&Go.x=16&Go y=14 302 0 341 611 40 80 HTTP/1.0
Mozilla/4.0+(compatible;+MSIE+5 01;+Windows+98;+TUCOWS)
SITESERVER=ID=22f117fb3708b2278f3c426796a78e2a;+ASPSESSIONIDQQGGQGPG=FCCCFIPBKJM
BDJJHBNCOEDGH http://ecommerce.com/Default.asp
2000-06-01 07:00:47 24 10 69 137 - W3SVC2 COOK_002 206 191.163.41 GET /directory/538 asp - 200 0 27471
459 881 80 HTTP/1.0 Mozilla/4.7+[en]+(Win98,+U)
ASPSESSIONIDQQGGQGPG=BHBCFIPBEJPNOMDPKCGLKNGC,
+ARSiteUser=1%2DC2B25364%2D3775%2D11D4%2DBAC1%2D0050049BD2E4,+ARSites=ALR=1
http://ecommerce.com/directory/34.asp
2000-06-01 07:00:47 207.136.48.117 - W3SVC2 COOK_002 206.191 163.41 GET /directory/511.asp - 200 0
77593 369 12538 80 HTTP/1.0 Mozilla/3.01Gold+(Win95;+I) ASPSESSIONIDQQGGQGPG=MFACFIPBDBN
PBFPBOENJKHJN;
+ARSiteUser=1%2DC2B251E5%2D3775%2D11D4%2DBAC1%2D0050049BD2E4;+ARSites=ALR=1
http://ecommerce.com/directory/506.asp
2000-06-01 07:00:49 192.102 216 101 - W3SVC2 COOK_002 206 191.163.41 GET /encyc/A1 asp ARRefSite=
15&ARRefCookie=1-C2B253B8-3775-11D4-BAC1-0050049BD2E4 200 0 47193 457 260 80 HTTP/1.0
Mozilla/4.7+[en]+(X11,+I,+SunOS+5.5.1+sun4u)
ASPSESSIONIDQQGGQGPG=PKBCFIPBIKONBPDHKDMMEHCE.http://ecommerce.com/hints/tips asp
```

Table 2 is an example portion of parser configuration data. The logical site definitions map a server IP address, port, and root URI to a logical site. For example, the entry "LOGICALSITEURIDEFINITION=209.114.94.26,80,/,1" maps all the accesses to port 80 of IP address 209.114.94.26 at URIs with a prefix "/" to logical site 1. The page type definitions map a logical site identifier, URI pattern, and query string pattern to a page type. For example, the entry "PAGEKEYDEFINITION=news item, news item, 1, {prefix}=homepage_include/industrynews_detail.asp, ,<NewsItemID>#{Uri}" indicates that a page type of "news item" is specified for logical site 1 by a URI pattern of "/homepage_include/industrynews_detail.asp." The definition also indicates that the event value is "<NewsItemID>#{Uri}," where the URI of the log entry is substituted for "{Uri} and the value of NewsItemID in the query string is substituted for "<NewsItemID>." The event type definitions map a site identifier, URI pattern, and query string pattern to an event type and value. The definitions also specify the name of the event type and the name of the dimension table for that event type. For example, the entry "EVENTDEFINITION=View News Article, View News Article, 1, {prefix}=/homepage_include/industrynews_detail.asp, <NewsItemId>=*, <NewsItemId>" indicates that View News Article event types are stored in the View News Article dimension table. That event type is indicated by a URI with "/homepage_include/industrynews_detail.asp," and the event value is the string that follows "<NewsItemId>=" in the query string.

TABLE 2

```
LOGICALSITEURIDEFINITION= 209.114.94.26, 80,/,1
PAGEKEYDEFINITION= news item, news item, 1, {prefix}=/homepage_include/industrynews_detail asp,,
<NewsItemId>#{Un}
PAGEKEYDEFINITION= page, page, 1,,, {Un}
EVENTDEFINITION= Login, Login, 1, {prefix}=/registration/login.asp,,
EVENTDEFINITION= Logout, Logout, 1, {prefix}=/registration/logout.asp,,
EVENTDEFINITION= Register Page 1, Register Page 1, 1, {prefix}=/registration/register.asp,,
EVENTDEFINITION= Register Page 2, Register Page 2, 1, {prefix}=/registration/register2.asp, <UserID>=*,
EVENTDEFINITION= Registration Confirmation, Registration Confirmation, 1,
{prefix}=/registration/register3.asp,,
EVENTDEFINITION= Abort Registration, Abort Registration 1, {prefix}=/registration/registrationabort.asp,,
EVENTDEFINITION= Member Services, Member Services, 1, {prefix}=/registration/memberservices.asp,,
EVENTDEFINITION= Change Password, Change Password, 1, {prefix}=/registration/changepassword.asp,,
EVENTDEFINITION= Profile Edit, Profile Edit, 1, {prefix}=/registration/profile.asp,,
```

TABLE 2-continued

EVENTDEFINITION= Change Affiliation, Change Affiliation, 1, {prefix}=/registration/changeaffiliation.asp, <UserID>=",
EVENTDEFINITION= Change Secret Question, Change Secret Question, 1, {prefix}=/registration/changesecretquestion.asp,,
EVENTDEFINITION= Forgot Information, Forgot Information, 1, {prefix}=/registration/forgotinfo.asp,,
EVENTDEFINITION= Forgot Password, Forgot Password, 1, {prefix}=/registration/forgotpassword.asp,,
EVENTDEFINITION= Forgot Signin, Forgot Signin, 1, {prefix}=/registration/forgotsignin.asp,,
EVENTDEFINITION= View News Article, View News Article, 1, {prefix}=/homepage_include/industrynews_detail.asp, <NewsItemId>=", <NewsItemId>

FIGS. **5–14** are flow diagrams of components of the parser in one embodiment. FIG. **5** is a flow diagram illustrating the parse log data routine that implements the main routine of parser in one embodiment. The routine processes each entry in the log file based on the parser configuration data. The routine filters out certain log entries, normalizes the attribute values of the log entries, and generates entries in the dimension tables for the attributes of the log entries. After processing all the log entries, the parser identifies user sessions and generates various statistics. In blocks **501–508**, the routine loops selecting and processing each log entry. In block **501**, the routine selects the next log entry of the log file starting with the first log entry. The routine may also pre-process the header information of the log file to identify the fields of the log entries. In decision block **1502**, if all the log entries have already been selected, then the routine continues at block **509**, else the routine continues at block **503**. In block **503**, the routine extracts the values for the fields of the selected log entry. In block **504**, the routine invokes the filter log entry routine, which returns an indication as to whether the selected log entry should be filtered out. In decision block **505**, if the filter log entry routine indicates that the selected log entry should be filtered out, then the routine skips to block **508**, else the routine continues at block **506**. In block **506**, the routine invokes the normalize log entry routine to normalize the values of the fields of the selected log entry. In block **507**, the routine invokes the generate dimensions routine to update the dimension tables based on the selected log entry and to add an entry into the log entry fact table. In block **508**, the routine updates the statistics for the log file. For example, the routine may track the number of log entries that have been filtered out. The routine then loops to block **501** to select the next log entry. In block **509**, the routine outputs the log file statistics. In block **510**, the routine invokes the identify sessions routine that scans the log entry table to identify the user sessions and updates a session dimension table. In block **511**, the routine invokes the generate aggregate statistics routine to generate various statistics and then completes.

FIG. **6** is a flow diagram of the filter log entry routine in one embodiment. The filter log entry routine is passed a log entry and determines whether the log entry should be filtered out. In blocks **601–607**, the routine determines whether the filter out conditions have been satisfied. In decision block **601**, the routine determines whether the log entry has a field count problem. A field count problem arises when the number of fields in the log entry does not correspond to the number of expected fields for that log entry. The number and types of fields may be defined in a "fields" directive line of the log file. In decision block **602**, the routine determines whether the log entry is outside of a specified time range. The routine compares the time field of the log entry to the time range. The time range may be specified so that only those log entries within that time range are processed. In decision block **603**, the routine determines whether the IP address of the log entry should be ignored. For example, a log entry may be ignored if the entry originated from a server whose function is to ping the customer's web server at periodic intervals. In decision block **604**, the routine determines whether the log entry corresponds to a comment (e.g., a "#remarks" directive). In decision block **605**, the routine determines whether the success code associated with the log entry indicates that log entry should be ignored. For example, if the success code indicates a failure, then the log entry may be ignored. In decision block **606**, the routine determines whether the log entry is requesting a resource whose extension indicates that the log entry should be ignored. For example, the routine may ignore log entries requesting graphic files, such as those in the ".gif" format. In decision block **607**, the routine determines whether the values within the fields of the log entry are corrupt. For example, a value in the date field that indicates a date of February 30th is corrupt. One skilled in the art would appreciate that the various filtering conditions may be specified in a configuration file. For example, the time range, IP addresses, and so on may be specified in the configuration file. These configuration files may be specified on a customer-by-customer basis.

FIG. **7** is a flow diagram illustrating the normalize log entry routine. The routine normalizes the values of the fields in the passed log entry. In block **701**, the routine converts the time of the log entry into a standard time such as Greenwich Mean Time. In block **702**, the routine corrects the time based on the variation between the times of the customer web servers. For example, the time of one web server may be five minutes ahead of the time of another web server. This correction may be based on current time information collected from computer systems that generated the events and then correlated to base current time information. In block **703**, the routine normalizes the values of the fields of the log entry. This normalization may include processing search strings to place them in a canonical form. For example, a search string of "back pack" may have a canonical form of "backpack." Other normalization of search strings may include stemming of words (e.g., changing "clothes" and "clothing" to "cloth"), synonym matching, and first and last word grouping. The first word grouping for the search strings of "winter clothing" and "winter shoes" results in the string of "winter."

FIG. **8** is a flow diagram of the generate dimensions routine in one embodiment. This routine identifies a value for each dimension associated with the passed log entry and ensures that the dimension tables contains entries corresponding to those values. In one embodiment, each entry in a dimension table includes the attribute value (e.g., user identifier) and a hash value. The hash value may be used by the loader when transferring information to the main data warehouse. Also, each entry has a local identifier, which may be an index into the local dimension table. The loader maps these local identifiers to their corresponding main identifiers

that are used in the main data warehouse. In block **801**, the routine invokes a routine that identifies the logical site associated with the log entry and ensures that an entry for the logical site is in the logical site dimension table. In block **802**, the routine invokes a routine that identifies the user associated with the log entry and ensures that an entry for the user is in the user dimension table. In block **803**, the routine invokes a routine that identifies the URI associated with log entry and ensures that an entry for that URI is in the URI dimension table. In block **804**, the routine invokes a routine that identifies the page type based on the parser configuration data and ensures that an entry for that page type is in the page type dimension table. In block **805**, the routine invokes a routine that identifies the various events associated with the log entry based on the parser configuration data and ensures that an entry for each event type is in the corresponding event table. In block **806**, the routine identifies other dimensions (e.g., referrer URI) as appropriate. In block **807**, the routine adds an entry to the log entry table that is linked to each of the identified dimensions using the local identifiers. In block **808**, the routine updates the statistics information based on the log entry and then returns.

FIG. **9** is a flow diagram of the identify logical site routine in one embodiment. This routine compares the site information of the passed log entry with the logical site definitions in the parser configuration data. In block **901**, the routine selects the next logical site definition from the parser configuration data. In decision block **902**, if all the logical site definitions have already been selected, then the routine continues the block **905**, else the routine continues at block **903**. In decision block **903**, if the URI of the log entry matches the selected logical site definition, then the routine continues at block **904**, else the routine loops to block **901** to select the next logical site definition. In block **904**, the routine updates the logical site dimension table to ensure that it contains an entry for the logical site defined by the selected logical site definition. The routine then returns. In block **905**, the routine updates the logical site dimension table to ensure that it contains a default logical site definition and then returns. The log entries that do not map to a logical site definition are mapped to a default logical site.

FIG. **10** is a flow diagram of the identify user routine in one embodiment. This routine may use various techniques to identify the user associated with the passed log entry. In one embodiment, the selection of the technique is configured based on the customer web site. For example, one customer may specify to use a cookie to identify users. In absence of a user identifier in the cookie, the industry norm is to identify users based on their IP addresses. This routine illustrates a technique in which a combination of cookies and IP addresses are used to identify a user. In block **1001**, the routine extracts the user identifier from the cookie associated with the log entry. The format of a cookie may be specified on a customer-by-customer basis. In decision block **1002**, if the extraction from the cookie was successful, then the routine continues at block **1006**, else the routine continues at block **1003**. The extraction may not be successful if, for example, the log entry did not include a cookie. In block **1003**, the routine extracts the IP address from the log entry. In decision block **1004**, if the IP address is determined to be unique, then routine continues at block **1006**, else the routine continues at block **1005**. Certain IP addresses may not be unique. For example, an Internet service provider may use one IP address for many of its users. The Internet service provider performs the mapping of the one IP address to the various users. In block **1005**, the routine extracts the browser identifier from the log entry. The combination of IP address and browser identifier may uniquely identify a user. In block **1006**, the routine updates the user dimension table to ensure that it has an entry for this user and then returns.

FIG. **11** is a flow diagram of the identify page type routine in one embodiment. This routine uses the page type definitions of the parser configuration data to identify the page type associated with the log entry. In block **1101**, the routine selects the next page type definition from the parser configuration data. In decision block **1102**, if all the page type definitions have already been selected, then no matching page type has been found and the routine returns, else the routine continues at block **1103**. In decision block **1103**, if the log entry matches the selected page type definition, then the routine continues at block **1104**, else the routine loops to block **1101** to select the next page type definition. In block **1104**, the routine updates the page type dimension table to ensure that it contains an entry for the page type represented by the selected page type definition. The routine then returns.

FIG. **12** is a flow diagram illustrating the identify events routine in one embodiment. This routine determines whether the log entry corresponds to any of the events specified in the parser configuration data. In block **1201**, the routine selects the next type of event from the parser configuration data. In decision block **1202**, if all the event types have already been selected, then the routine returns, else the routine continues at block **1203**. In block **1203**, the routine selects the next event definition of the selected event type. In decision block **1204**, if all the event definitions of the selected event type have already been selected, then the log entry does not correspond to this type of event and the routine loops to block **1201** to select the next type of event, else the routine continues at block **1205**. In block **1205**, if the log entry matches the selected event definition, then the routine continues at block **1206**, else the routine loops to block **1203** to select the next event definition of the selected event type. In block **1206**, the routine updates the dimension table for the selected type of the event to ensure that it contains an entry for the selected event definition. The routine then loops to block **1201** to select the next type of event. In this way, the routine matches no more than one event definition for a given event type. For example, if there are two event definitions for the event type "Keyword Search," then if the first one processed matches, then the second one is ignored. Those skilled in the art will appreciate that in other embodiments each event definition could be checked for a match. Similarly, in other embodiments only a single event may be matched for each log entry, or multiple page type definitions may be matched for each log entry.

FIG. **13** is a flow diagram illustrating the identify sessions routine in one embodiment. This routine scans the log entry table of the local data warehouse to identify user sessions. In one embodiment, a user session may be delimited by a certain period of inactivity (e.g., thirty minutes). The criteria for identifying a session may be configurable on a customer-by-customer basis. In block **1301**, the routine selects the next user from the user dimension table. In decision block **1302**, if all the users have already been selected, then the routine returns, else the routine continues at block **1303**. In block **1303**, the routine selects the next log entry for the selected user in time order. In decision block **1304**, if all log entries for the selected user have already been selected, then the routine loops to block **1301** to select the next user, else the routine continues at block **1305**. In decision block **1305**, if the selected log entry indicates that a new session is starting (e.g., its time is more than 30 minutes greater than that of the last log entry processed), then the routine con-

tinues at block **1306**, else the routine loops to block **1303** to select the next log entry for the selected user. In block **1306**, the routine updates a session fact table to add an indication of the new session. The routine then loops to block **1303** to select the next log entry for the selected user. The routine may also update the log entries to reference their sessions.

FIG. **14** is a flow diagram of the generate aggregate statistics routine in one embodiment. This routine generate statistics based on analysis of the fact and dimension tables used by the parser. In block **1401**, the routine selects the next fact table of intent. In decision block **1402**, if all the fact tables have already been selected, then the routine returns, else the routine continues at block **1403**. In block **1403**, the routine selects the next entry of the selected fact table. In decision block **1404**, if all the entries of the selected fact table have already been selected, then the routine loops to block **1401** to select the next fact table, else the routine continues at block **1405**. In block **1405**, the routine aggregates various statistics about the selected fact table. The routine then loops to block **1404** to select the next entry of the fact table.

FIGS. **15–17** are flow diagrams illustrating components of the loader in one embodiment. FIG. **15** is a flow diagram of the load log data routine implementing the main routine of the loader in one embodiment. This routine controls the moving of the data from the local data warehouse (created and used by the parser) into the main data warehouse. In block **1501**, the routine invokes the create partitions routine to create partitions for the main data warehouse as appropriate. In blocks **1502–1504**, the routine loops loading the dimension tables into the main data warehouse. In block **1502**, the routine selects the next dimension table. In decision block **1503**, if all the dimension tables have already been selected, then the routine continues at block **1505**, else the routine continues at block **1504**. In block **1504**, the routine invokes the load dimension table routine for the selected dimension table. The routine then loops to block **1502** to select the next dimension table. In blocks **1505–1507**, the routine loops adding the entries to the fact tables of the main data warehouse. In block **1505**, the routine selects the next fact table in order. The order in which the fact tables are to be loaded may be specified by configuration information. The fact tables may be loaded in order based on their various dependencies. For example, a log entry fact table may be dependent on a user dimension table that is itself a fact table. In decision block **1506**, if all the fact tables have already been loaded, then the routine returns, else the routine continues at block **1507**. In block **1507**, the routine invokes the load fact table routine for the selected fact table. The routine then loops to block **1505** to select the next fact table.

FIG. **16** is a flow diagram of the load dimension table routine in one embodiment. This routine maps the local identifiers used in the local data warehouse to the main identifiers used in the main data warehouse. In block **1601**, the routine selects the next entry from the dimension table. In decision block **1602**, if all the entries of the dimension table have already been selected, then the routine returns, else the routine continues at block **1603**. In block **1603**, the routine retrieves an entry from the dimension table of the main data warehouse corresponding to the selected entry. In decision block **1604**, if the entry is retrieved, then the routine continues at block **1606**, else the dimension table does not contain an entry and the routine continues at block **1605**. In block **1605**, the routine adds an entry to the dimension table of the main data warehouse corresponding to the selected entry from the dimension table of the local data warehouse. In block **1606**, the routine creates a mapping of the local identifier (e.g., index into the local dimension table) of the selected entry to the main identifier (e.g., index into the main dimension table) for that selected entry. The routine then loops to block **1601** to select the next entry of the dimension table.

FIG. **17** is a flow diagram of the load fact table routine in one embodiment. This routine adds the facts of the local data warehouse to the main data warehouse. The routine maps the local identifiers for the dimensions used in the local warehouse to the main identifiers of dimensions used in the main data warehouse. In block **1701**, the routine selects the next entry in the fact table. In decision block **1702**, if all the entries of the fact table have already been selected, then the routine returns, else the routine continues at block **1703**. In block **1703**, the routine selects the next dimension for the selected entry. In decision block **1704**, if all the dimensions for the selected entry have already been selected, then the routine continues at block **1706**, else the routine continues at block **1705**. In block **1705**, the routine retrieves the main identifier for the selected dimension and then loops to block **1703** to select the next dimension. In block **1706**, the routine stores an entry in the fact table of the main data warehouse. The routine then loops to block **1701** to select the next entry in the fact table.

FIG. **18** is a flow diagram illustrating the identify user aliases routine in one embodiment. This routine tracks the different user identifiers as a user switches from one web site to another. In particular, the routine maps the user identifiers used by a referrer web site to the user identifiers used by the referred-to web site. In this way, the same user can be tracked even though different web sites use different identifiers for that user. This routine may be invoked as part of the parsing of the log files. In decision block **1801**, if the log entry indicates a referrer web site, then the routine continues at block **1802**, else the routine returns. In block **1802**, the routine identifies the user identifier for the referrer web site. In block **1803**, the routine creates a mapping between the referrer user identifier and the referred-to user identifier. The routine then returns.

As noted above, interaction data (e.g., navigation data from interactions by users with a customer's web site) can be analyzed by the parser component to identify various occurrences of interest. In particular, the parser component uses parser configuration data (also referred to as "data parsing information") that defines various types of occurrences so that any such occurrences in the interaction data can be identified. For example, when analyzing a customer's web site interaction data, the parser component can use data defining customer-specific categories of web pages (e.g., web pages with shoe product information) and customer-specific web site events of interest (e.g., when users of the customer's web site search for product information or add an item to their shopping cart). Such high-level types of occurrences can be specified in a variety of ways, such as by using a combination of a logical web site, one or more URIs corresponding to web pages, and/or one or more query strings. The parser configuration data may also specify a mapping of actual web sites to one or more logical sites, as well as event-specific information to be extracted from the interaction data and stored in the data warehouse.

FIGS. **19A–19AE** illustrate various example user interactions with an example web site www.digimine.com for digiMine that has various web pages, and Tables 3–6 illustrate various examples of data parsing information that corresponds to the web site. Those skilled in the art will appreciate that these web pages and types of interactions are

merely examples, and that in other embodiments various types of interaction or usage data related to a wide variety of types of content sets (e.g., interactions with or use of a web-based or telecommunications-based service, interactions with or use of an executing computer program or a device, etc.) can instead have data parsing information that is used for analysis of the data.

In particular, FIG. **19**A illustrates an example web page **1900** that is displayed at a client computer after a user specifies the URI www.digimine.com for the digiMine web site to an executing web browser program on the client. The web page includes various informational content **1910**, and various user-selectable controls including controls **1901–1909**. As is discussed in greater detail below, the web site has several sections that each contain distinct related types of information, and controls **1903**, **1905**, **1907**, and **1909** can be used to obtain an overview web page for each of four different sections. Control **1904** is an alternate method by which the user can obtain the overview web page for the "Services" section of the web site (also accessible via control **1903**), and control **1901** causes the currently displayed web page **1900** to be displayed. Those skilled in the art will appreciate that this web page is sent to the client computer by a web server for the digiMine web site, and that an entry corresponding to this interaction (i.e., a request for the web page corresponding to the specified URI) will typically be added to a log file for that web server.

If the user interacts with the web site to select control **1903** (or control **1904**), the web page illustrated in FIG. **19**B will be sent to the client computer and displayed to the user. As previously noted, this web page is an overview for the Services section of the web site, and it includes various informational content **1915** related to services provided by digiMine to its customers. The web page also includes the same controls **1901**, **1903**, **1905**, **1907**, and **1909** as did web page **1900**. In addition, the currently displayed web page also includes other controls **1912**, **1914**, and **1916–1928**. Control **1922** causes the currently displayed web page to be displayed, and the other newly displayed controls cause other web pages to be displayed that contain additional detailed information within the Services section of the web site.

If the user interacts with the web site to select control **1912** (labeled "digiMine Warehousing Services"), the web page illustrated in FIG. **19**C will be displayed to the user. As with the previously displayed web pages, this web page includes various informational content as well as many of the same controls as the web page illustrated in FIG. **19**B. As is shown by indication **1920**, this web page has a corresponding URL of "www.digimine.com/services/warehousing.htm." As would be expected based on the label for control **1912** and the text portions of the URL path for the page, this web page includes informational content related to data warehousing services that digiMine provides to customers.

In a similar manner, if the user interacts with the web site to select control **1914** displayed on the web page illustrated in FIG. **19**C (or on the web page illustrated in FIG. **19**B), the web page illustrated in FIG. **19**D will be displayed to the user. As would be expected, the displayed web page includes informational content related to data analysis services provided by digiMine, and also includes various controls. When the control **1916** is selected, the web page illustrated in FIG. **19**E is displayed, and selection of the control **1918** causes the web page illustrated in FIG. **19**F to be displayed.

Rather than corresponding to web pages containing detailed information about specific types of provided services, controls **1924**, **1926** and **1928** instead correspond to web pages containing other higher-level information about provided services. In particular, selection of control **1924** causes the web page illustrated in FIG. **19**G to be displayed, with the web page discussing various benefits to a customer from the various provided services. Similarly, selection of control **1926** causes the web page illustrated in FIG. **19**J to be displayed, and selection of control **1928** causes the web page illustrated in FIG. **19**K to be displayed.

Several of the web pages from the Services section of the web site also include a control **1930** that corresponds to a detailed Data Sheet related to the digiMine services. While the previously displayed web pages have been specified in HTML format, the Data Sheet is a PDF document that is illustrated in FIGS. **19**H and **191**. The web pages and PDF document illustrated in FIGS. **19**B–**19**K are the web pages that are part of the Services section of the digiMine web site in this example embodiment.

If the "Company" section control **1905** is instead selected from any of the previously displayed web pages, an overview of the company will be presented to the user in the web page illustrated in FIG. **19**L. FIGS. **19**L–**19**Q illustrate some of the web pages that are part of the Company section of the digiMine web site in this illustrated embodiment. In addition to the top-level controls **1901**, **1903**, **1905**, **1907**, and **1909**, the illustrated web page also includes Company section-specific controls **1931–1939**. For example, if control **1933** is selected, the web page illustrated in FIG. **19**M will be displayed containing information about the management team for the company. This web page includes controls **1941–1949** corresponding to different members of the management team, and selection of control **1949**, for example, displays the web page illustrated in FIG. **19**N related to the Vice President of Legal Affairs, Bob Bolan.

The various sections of the web site can include various subsections in a hierarchical manner, and any such subsection can similarly contain its own hierarchical subsections. For example, the "Careers" subsection of the Company section of the web site can be accessed by selecting control **1937**. In response, the web page illustrated in FIG. **190** will be displayed in which various overview information about working at digiMine is presented. Various controls are available to obtain additional web pages from the Careers subsection of the Company section, such as controls **1950** and **1953**. Selection of the control **1950** causes the web page illustrated in FIG. **19**P to be displayed, in which the Careers subsection is separated into additional subsections based on the types of available jobs as is shown by controls **1951**. Selection of the "Legal" control **1952** causes the web page illustrated in FIG. **19**Q to be displayed. In the illustrated embodiment, the URL indications **1920** for the various displayed web pages contain information that reflects the hierarchical nature of the sections and subsections of the web site. For example, the URL **1920** illustrated in FIG. **19**O shows that the file structure for the web page includes a "careers" hierarchy member that is one hierarchy level below a "company" hierarchy member, which is at a first hierarchy level for the digiMine web site. Those skilled in the art will appreciate that in some embodiments each hierarchy member may reflect a hierarchical manner of storing the associated web pages or other information, such as by having a "careers" directory that is a subdirectory of a "company" subdirectory, which is itself a subdirectory of the digiMine web site.

If the control **1907** is selected on any of the previously displayed web pages, an overview web page for the "Media Center" section of the web site will be displayed, as is

illustrated in FIG. 19R. As is shown, subsections of the web site corresponding to press releases or to news articles can be accessed by selecting the displayed controls **1959** and **1957** respectively. After the "press releases" control **1959** is selected, the web page illustrated in FIG. 19S is displayed, with controls **1956** indicating various press releases that are available from this subsection of the web site.

If the control **1909** is selected on one of the previously displayed web pages, the "Customer Log In" web page illustrated in FIG. 19T is displayed in response. As is shown, this web page includes a user-editable portion **1960** in which customers can interact with the web site in a manner other than merely selecting controls, such as by specify appropriate customer-specific access information in the appropriate form fields in order to obtain access to data for their own web site. In addition, as is shown by URL **1920**, the customer-specific section of the digiMine web site is provided by a server using a different third-level domain name (i.e., insight.digimine.com) than the previously discussed sections of the web site (that use the third-level domain name www.digimine.com). Those skilled in the art will appreciate that this distinct third-level domain name may correspond to one or more web server machines that are distinct from the one or more web servers that support the www.digimine.com domain name, or that there may instead be partial or complete overlap in the respective web server machines. In addition, in the illustrated embodiment the web pages for the Customer Log In section of the web site are transmitted in a secure manner to protect confidential customer data (e.g., by using secure HTTP ("HTTPS") and a different port number than the standard port number **80** for unsecure HTTP).

In the illustrated embodiment, a user digimineqa from the Quality Assurance department of digiMine provides the appropriate access information on the web page illustrated in FIG. 19T and, after interacting with the web site by selecting the "submit" button, receives the web page **1972** illustrated in FIG. 19U. This web page is shown displayed within a web browser display window **1970**. The displayed web page includes multiple frames that are each able to display different content, including a control frame **1979** with various user-selectable controls **1977** and display frames **1975** in which customer-specific information is displayed. In the illustrated embodiment, the URL indication **1920** corresponds to the information displayed in the display frames. The path portion of the indicated URL specifies an executable Active Server Page ("ASP") program on the server that will supply the content displayed in the display frames, and the indicated URL also includes a query string portion that will be supplied as input to the executable program. In addition, note that in the illustrated embodiment, each customer receives a unique customer ID, and each customer's data is treated as a separate hierarchical section of the web site. For example, the ID for the current user is "I0033," which is shown in the hierarchy structure of the path portion of the URL. Those skilled in the art will appreciate that in other embodiments different customer data could instead be accessed in a variety of other ways, such as by using the same URL path for each customer for a given type of data but using differing query strings to identify the current customer (e.g., "customerID=I0033").

As is shown in FIG. 19U, a variety of types of information is available to each user, including administrative information related to the customer's account and information related to analysis of interaction or usage information from the customer. Those skilled in the art will appreciate that in some embodiments the analysis will have previously been performed and the analysis reports will use the information from the previous analysis (e.g., stored information), and in other embodiments the analysis can be dynamically performed when a report is requested by a customer.

In the web page illustrated in FIG. 19U, the user has interacted with the web site to select the "Users" control **1980** in the "Management Desk" section of the customer-selectable controls, with the display frames correspondingly containing administrative information about the users defined for the current customer. In the illustrated embodiment, there is a single "Administrators" user group defined, and a single user "digimineqa" (whose information was used in the customer login screen illustrated in FIG. 19T) that is a member of that user group. Those skilled in the art will appreciate that other customers may have multiple user groups defined, as well as having multiple users in one or more of their user groups. Note also that the "x" in the box next to the Users control **1980** indicates that it is the currently selected customer control. FIG. 19V illustrates a web page corresponding to an alternate user selection from the Management Desk section of the customer controls, that being the "Post Message" control **1981**. The display frame **1975** indicates that the current user can post a message that will be shown to other users. The URL indication **1920** for the display frame in this web page shows that a different ASP is specified to supply the displayed message form, and that the same query string as was used for the Users display is specified.

In addition to the administrative controls in the Management Desk section, there are a variety of data reports of differing types available to the user. The display frame illustrated in FIG. 19W contains an "Executive Summary" display for the user, as shown by selection of the "Executive Summary" control **1982**. The content of the display frame includes various groups of information such as a date range filter **1997**, a data chart **1995**, a data table **1993** (not shown in the currently scrolled position of the display frame), and a message window **1999** (also not shown in the currently scrolled position of the display frame). In addition, the display frame includes display controls **1992**, **1994**, **1996**, and **1998** with which the user can select whether to show or hide the various corresponding groups of information. The user can also modify the displayed information in various ways, such as by interacting with the web site to modify the specified date information in the date filter using the user-selectable controls and by interacting with the web site to alter the visual appearance of the chart or the data displayed in the chart via the various user-selectable display controls available within the display chart group of information.

In addition to the Executive Summary report, the "Reports" section of the customer controls includes groups of "Site Traffic" sub-section controls, "Site Usage" sub-section controls, "Customer" sub-section controls, "Data Mining" sub-section controls, and "Products and Transactions" sub-section controls. FIG. 19X illustrates a web page whose display frame includes a report corresponding to the "Hourly Activity" Site Traffic control **1983**. As with the Executive Summary report, the Hourly Activity report includes a date range filter, chart, table, and message window. As shown, other Site Traffic reports include a Daily Activity report, a Page Views per Visit report, a Frequently Viewed Pages report, and an Entry Path Summary report.

The Site Usage reports include a Visit Duration per User report, a Referring URL report, a Keywords Searched report, a Category Analysis report, an Event Analysis report, and a Funnel report. FIG. 19Y illustrates a web page whose display frame shows a Referring URL report, as indicated by

the selection of the Referring URL control **1984**. Conversely, FIG. **19**Z illustrates a web page whose display frame includes a Category Analysis report. In the illustrated display frame, the data table **1993** and message window **1999** are visible, and the data chart is currently hidden. The Category Analysis report provides various information for each of one or more categories, such as the number of Page Views for web pages of the category and the number of Unique Users who have viewed web pages of that category. In the illustrated embodiment, only the top-level categories are currently shown (as illustrated by the user-selectable control **1963**), with only a single top-level category currently defined for the digimineqa customer.

Those skilled in the art will appreciate that other users may have multiple top-level categories, and that the categories whose information is to be displayed can be selected in various ways. For example, all of the categories at all of the hierarchy levels could be displayed, and the user could then pick and choose any categories in which they have an interest. Alternately, a user could select a level of categories, such as top-level or second-level categories, and have information displayed for each category at that selected level. In other situations, it may be useful to display category information for a specified category and all sub-categories or super-categories in a hierarchical arrangement. Those skilled in the art will appreciate that categories to be displayed can be selected in other similar ways. FIG. **19**AA illustrates one example embodiment of displaying multiple categories for selection. As is shown, in the illustrated embodiment the categories are arranged in a hierarchical manner, thus allowing various groupings of categories to be chosen such as individual categories, all categories in a hierarchical structure, all categories at a specified level of the hierarchy, etc.

FIG. **19**AB illustrates a web page whose display frame includes an Event Analysis report, as indicated by the selection of the Event Analysis control **1986**. In the illustrated embodiment, only a single event type has been selected to have information displayed, that being the "Contact Form" event type **1964** (e.g. corresponding to each person that has interacted with the web site to request the web page corresponding to digiMine's contact form or to submit a completed contact form). As is shown, a variety of types of information can be illustrated for each event type, such as "Total Occurrences," "Unique Users," and "Occurrences per Visit," and information can be simultaneously displayed for multiple related or unrelated event types. Those skilled in the art will appreciate that event types whose information is to be displayed can be selected in a variety of ways, such as in a manner analogous to those discussed above with respect to multiple categories. FIG. **19**AC illustrates a Funnel report that provides one example of displaying information for multiple related event types, those being a sequence of related event types.

In addition to providing information about each of multiple categories individually, various types of information about the interactions of multiple categories can also be displayed. For example, the display frame of the web page illustrated in FIG. **19**AD shows a Category Affinity report in which information is provided about users that access web pages in each of the displayed categories in a single user session. Those skilled in the art will appreciate that categories to be included in such a report can be chosen in a variety of ways, such as was discussed previously for the Category Analysis report. Those skilled in the art will also appreciate that a variety of other types of similar information can be shown rather than merely combinations of categories, such as sequences of categories in which the order of the viewing is relevant. Similarly, in other embodiments affinity reports could be presented for other types of information, such as specified event types or combinations of categories and event types. FIG. **19**AE illustrates that, in addition to displaying various reports, information that is not customer-specific can also be provided, such as a glossary of terms. Those skilled in the art will appreciate that various other types of information can similarly be provided.

As previously noted, Tables 3–6 contain example data parsing information that can be used by the parser component to identify various high-level types of occurrences for the example digiMine web site illustrated in FIGS. **19**A–**19**AE. In some embodiments, occurrence types can be specified by using a web site or web server identifier, an identifier for one or more URIs, and/or one or more query string identifiers. Correspondingly, Tables 3–6 contain example data parsing information corresponding to identifying those types of information.

In particular, Table 3 contains example data parsing information used to identify the digiMine web site and its web servers. As previously illustrated in Table 1, each log entry to be parsed will typically include an IP address and a port number that are used to communicate with (e.g., send requests to) a web server computer.

The identification of whether a particular log entry corresponds to a particular web site is complicated by several factors. For example, it is common for web sites to use a primary domain name (e.g., www.digimine.com) whose corresponding IP address is a load balancing device that can direct client requests to multiple physical web server machines that each have their own distinct IP addresses. Thus, there will typically be multiple IP addresses for multiple web servers that can provide the same web pages for a web site. In some situations, all of the web servers for a web site will maintain a single log file for the entire web site, while in other situations each of the web servers will maintain a separate log. However, even if each web server maintains a separate log, in some situations the various log files will be combined together before they are processed by the parser component. Thus, each entry in the log file can correspond to different physical machines that are acting as web servers for the web site.

In addition to having multiple alternate web servers that can each provide any of the web site content, in other situations a web site may have certain subsections or types of processing (e.g., server-executed code) that are provided by one or more web servers that are distinct from the other web servers providing the rest of the content for the web site. In these situations, communications shown in the log file that are directed to those web servers will typically be restricted to those portions of the web site or types of processing handled by the web servers.

In addition to having multiple web servers that each provide some or all of the content for a web site, in other situations a single machine will act as a web server for multiple web sites. In such situations, each web site can have a distinct domain name that may be mapped to a distinct IP address, but all of the IP addresses refer to that single physical machine. In such a situation, if the machine maintains a single log file for any requests that it receives, then the log file will contain entries for each of the web sites that it hosts. Thus, in such a situation it is useful to be able to determine the log entries that correspond to a particular web site of interest.

In the example site data parsing information illustrated in Table 3 below, it can be seen that the digiMine web site is